THIS MANUAL IS CURRENT
FOR MDBS 3.09

MDBS III

DATA BASE DESIGN REFERENCE MANUAL

THE MDBS DDL MANUAL

VERSION 3.09

MARCH 1987

Micro Data Base Systems, Inc.

P. O. Box 248

Lafayette, Indiana 47902

USA

Telex: 9102401559 WLAF UQ

(800) 344-5832

Copyright Notice

This entire manual is provided for the use of the customer and the customer's employees. The entire contents have been copyrighted by Micro Data Base Systems, Inc., and reproduction by any means is prohibited except as permitted in a written agreement with Micro Data Base Systems, Inc.

© 1987 Micro Data Base Systems, Inc.

3250921

NEW VERSIONS, LICENSING, AND DISCLAIMER

Any programming endeavor of the magnitude of the MDBS III software continually evolves over time. Realizing this, Micro Data Base Systems, Inc. (**mdbs**), provides updates to this product as part of a service and maintenance program.

New versions of MDBS III software are considered separate products; however, bona fide owners of previous versions are generally entitled to a preferential rate structure.

Each copy of our software is personalized to identify the licensee. There are several levels of this personalization, some of which involve encryption methods guaranteed to be combinatorially difficult to decipher. Our products have been produced with a very substantial investment of capital and labor, to say nothing of the years of prior involvement in the data base management area by our principals. Accordingly, we are seriously concerned about any unauthorized copying of our products and will take any and all available legal action against illegal copying or distribution of our products.

All rights reserved. No part of this material shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from **mdbs**.

Although care has been taken in the preparation of this material, **mdbs** assumes neither responsibility for errors or omissions, nor for damages resulting from the use of the information contained herein. **mdbs** does not warrant its accuracy nor guarantee the operation of the system in every instance described herein.

The reader/user assumes full liability and all risk for any damages resulting from the use of the information contained herein, and for determining whether the information contained herein is suitable for user's intended purpose.

mdbs reserves the right to incorporate design improvements and new functions in its software products and software systems. Recent improvements may not always be reflected in documentation.

© Copyright 1987 Micro Data Base Systems, Inc.

PREFACE

This manual provides an introductory overview of application development with MDBS III. It then concentrates in detail on the features and utilization of the MDBS III Data Description Language. Companion manuals discuss the MDBS III Data Manipulation Language and various optional modules. For a tutorial treatment of data base management (explaining its value, characteristics, and advantages), consult such suitable references as:

- Micro Database Management - Practical Techniques for Application Development by R.H. Bonczek, et. al., 536 pages, Academic Press, New York, 1984.
- "A Perspective on Data Models," PC Tech Journal, Vol. 2, No. 1, 1984.
- "Micros Get Mainframe Data Scheme," Systems and Software, Vol. 3, No. 5, 1984.
- "Uniting Relational and Postrelational Database Management Tools", Systems and Software, Vol. 3, No. 11, 1984.

The first reference provides the most definitive coverage of the postrelational approach, comparing and contrasting it with the four older data base management approaches. That book also includes extensive examples of MDBS III usage. It is available through your local bookstore, from the publisher, or from MDBS, Inc.

The MDBS commitment to customer success does not stop with software innovation, quality and support. The company offers a full range of consulting and application development services to clients from regional offices located throughout the United States and Canada. Here are some of the services MDBS provides:

- Customized application systems design
- Customized application systems development using GURU, KnowledgeMan and/or the MDBS III tools
- Communications interfaces including mainframe-micro links
- Training seminars for MDBS III, KnowledgeMan, and GURU in major cities and at customer sites
- Special training on a wide variety of topics including specific application systems, C programming, MDBS III usage, and GURU usage
- General consulting including feasibility studies and hardware/software recommendations

MDBS, KnowledgeMan, and GURU are registered trademarks of Micro Data Base Systems, Incorporated.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL

TABLE OF CONTENTS

Page

PREFACE

CHAPTER I. OVERVIEW I-1
Introduction to MDBS III I-2
MDBS Software Modules I-3
Manual Organization I-5
Data Base Design I-6
DDL Analyzer I-8
DBRS I-9
Access Languages I-9
DML I-10
IDML I-11
QRS I-12
RDL I-12
BLF I-12
Data Integrity Facilities I-13
Data Security Facilities I-14
Data Base Users I-14
Data Encryption I-14
Performance Tuning Facilities I-14
Multiuser Processing I-15
MDBS III and KnowledgeMan: Decision Support I-16
MDBS III and Guru: Expert Systems I-18
MDBS Data Description Language and Software Summary I-19

CHAPTER II. DEFINITIONS AND EXAMPLES II-1
Data Base Areas and Pages II-1
The Logical Structure of an MDBS Data Base II-4
Set Ordering II-9
Special Ways of Using Sets II-11
Set Retention II-14
Set Insertion (Connection) II-14
Types and Sizes of Data Items II-14
Controlling Record Placement II-17
Data Security II-19
Miscellaneous II-19

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL

CHAPTER III. MDBS.DDL SPECIFICATIONS III-1
Notational Conventions III-1
DDL Sections III-6
Comments III-7
Identification Section III-8
User Section III-13
Area Section III-14
Record Section III-17
Data Item Section III-21
Set Section III-24
Owner Section III-26
Member Section III-29
End Section III-31

CHAPTER IV. DATA SECURITY IV-1
Passwords IV-1
Encryption IV-1
Access Codes IV-1
DDL Specification of Access Codes IV-3

CHAPTER V. DATA INTEGRITY V-1
Preventing Unauthorized Changes V-1
Fixed Set Retention V-1
Duplicate Key Exclusion V-2
Invalid Dates and Times Prohibited V-2
Feasibility Range Specification V-2
Data Base Recovery and Transaction Logging V-5

CHAPTER VI. USING THE MDBS.DDL SOFTWARE VI-1
Invoking MDBS.DDL VI-2
Command Line Arguments (global) VI-2
Command Line Arguments (batch mode) VI-3
Interactive MDBS.DDL Use VI-3
Special Keys VI-5
Interactive Commands VI-6
Help VI-7
Text Entry (and Deletion) VI-7
Line Editing VI-14
Global Editing VI-20
DDL Analysis VI-22
Stop VI-25

CHAPTER VII. DDL ANALYZER MESSAGES VII-1

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL

Page

FIGURES
I-1. MDBS Software Modules I-3
I-2. Examples of the Three Kinds of Relationships between a Pair of Record Types I-7
I-3. Examples of the Three Kinds of Recursive Relationships Involving a Single Record Type I-7
I-4. Examples of Some Forked Relationships I-7
I-5. Using the MDBS.DDL Module I-8
I-6. Using the MDBS.DMS Module I-10
II-1. Areas and Pages II-3
II-2. A Sample Logical Structure II-5
II-3. System-owned Sets II-11
II-4. A Recursive Set II-12
II-5. Multiple Member Set II-12
II-6. Multiple Owner Set II-13
III-1. Sample DDL III-2
III-2. Sample DDL-Advanced III-3
III-3. Notation III-5

TABLES
II-1. Types of Data Items II-2
IV-1. MDBS Access Privileges IV-2
V-1. MDBS Data Item Feasibility Range Limits V-3
VI-1. MDBS.DDL Interactive Commands VI-6

APPENDICES
APPENDIX A. Data Description Language Keywords A-1
APPENDIX B MDBS Data Description Language Syntax B-1
APPENDIX C. Maximums and Minimums in a DDL Source Specification ..... C-1 to C-2
APPENDIX D. Defaults D-1
APPENDIX E. Data Item Range Examples E-1
APPENDIX F. Estimating Data Base Sizes F-1
APPENDIX G. List of QRS Keywords G-1
APPENDIX H. Alternative Layout for DDL Specification of Figure III-2 ..... H-1

INDEX Index-1

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

I. OVERVIEW

This **MDBS DDL Manual** is the first component in the set of MDBS III Reference Manuals. It describes the data structuring facilities available for designing MDBS data basesand the Data Description Language (DDL) for formally specifying data base designs.

The second component in the manual set is the **MDBS DMS Manual**, which describes data manipulation commands available to application programmers. Beyond these two basics, other manuals in the set describe various MDBS software options.

The MDBS III Reference Manuals discuss the aspects of MDBS usage common to all systems, regardless of the host hardware, operating system, and programming language. These items include the DDL features and syntax, the Data Manipulation System (DMS) commands, and query command languages.

The **MDBS System Specific Manuals** describe the environment dependent features of MDBS, such as operating procedures and the DMS command syntax of the host programming language. MDBS varies slightly from one operating environment to another, due to differences in hardware, operating systems, host languages, and the version of MDBS.

Two versions of the MDBS III data base control system are available to operate on a wide variety of hardware.

- MDBS Version 3a operates on the IBM PC series and compatibles running the PCDOS or MSDOS operating system. Version 3a is written in assembly language to optimize the performance of the data base control system.

- MDBS Version 3c operates on hardware such as the Intel 286/310, 68000-based machines, AT&T 3B2-3B5, DEC VAX 11/780 machines and MicroVAX/II running operating systems such as VMS, UNIX, XENIX and ULTRIX. Version 3c is written in the C language.

The MDBS III data structuring features presented in this manual divide into two levels: fundamental and advanced. An understanding of the fundamental features is sufficient for designing many application systems using MDBS III.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

The advanced DDL features are additional mechanisms to control the characteristics and performance of the data base. These features provide the application developer with greater flexibility and cover several areas:

- controlling record placement within the data base
- segmenting a data base into areas (several physical files)
- controlling data access (data security)
- controlling data item values (feasibility range checking)
- tuning performance

INTRODUCTION TO MDBS III

MDBS III is a powerful data base management system (dbms) designed to maximize the productivity of professional application developers. MDBS III addresses the major issues facing original equipment manufacturers (OEMs), value added resellers (VARs), and information services groups:

- quality application system software
- reduced development time and cost
- effective administration of application systems
- data sharing among application systems and generic decision support software
- portable application software for multiple operating environments

MDBS III provides facilities for enforcing data security, ensuring data integrity, automatic program generation, concurrency control and performance tuning. These facilities are important for productive application development and on-going effective administration.

Subject to its integrity and security controls, MDBS III promotes data sharing in both local area network and multiuser operating system settings. The same data base can simultaneously be accessed by a variety of application programs whose host languages may vary from COBOL to C. It can be directly accessed by expert systems using the GURU inference engine. It can also be directly accessed through end user facilities such as the nonprocedural query interface and the KnowledgeMan/2 decision support environment.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

MDBS SOFTWARE MODULES

There are two essential MDBS III software modules: the DDL Analyzer (MDBS.DDL) and the Data Base Control System (MDBS.DMS). Other modules are optional; their availability varies for each operating environment.

The Data Description Language Analyzer (MDBS.DDL) builds new data bases, each consisting of a data dictionary and several initialized, empty pages for data storage. The designer specifies a data base structure with the Data Description Language (DDL). Once you create a data base, the data base control system can process Data Manipulation Language (DML) commands to create, retrieve, and modify data in the data base. These commands can originate from an application program or one of the MDBS optional modules.

One optional module is the Recovery and Transaction Logging form (MDBS.RTL) of the data base control system. This alternative form of MDBS.DMS provides all the capabilities of the standard data base control system; however, MDBS.RTL includes additional functions to record all data base changes (transactions) in a logging file. MDBS.RTL is furnished with a recovery utility for processing transaction logs. The **MDBSIII RTL Manual** describes the recovery and transaction logging facilities.

In addition to the DML, other optional software modules can access the data base. Each module uses the Data Base Control System to carry out data base access requests. Optional modules exist for batch loading a data base, interactive DML processing, English-like queries, customized report generation, linking the data base with a decision support environment, and inference engine processing of data base contents. Optional modules are also available to help data base administrators and developers modify or restructure existing data bases to conform with new designs. Figure I-1 shows how these software modules interact with the data base.

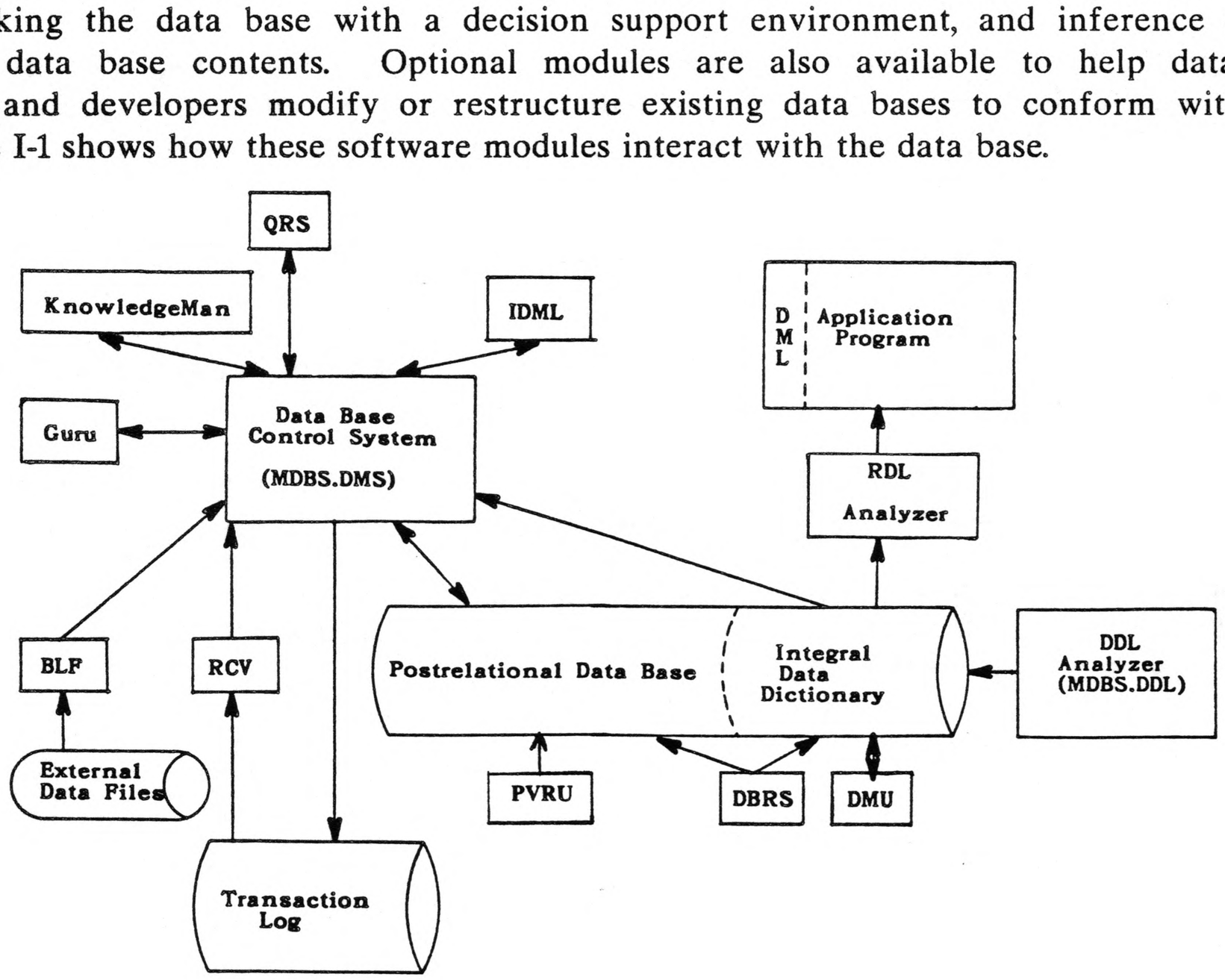


Figure I-1. MDBS Software Modules

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

The MDBS III modules in figure I-1 include these functions:

BLF - the Batch Load Facility that automatically loads the contents of an external file into an extended network architecture (ENA) data base.

Data Base Control System - the dictionary-driven Data Base Control System that controls all access to the encrypted ENA data base, controls all aspects of data sharing among concurrent users, and (for the RTL form) automatically logs all transactions.

DBRS - the Data Base Restructuring System that a data base administrator uses to revise the schema of an existing data base and to re-optimize physical storage utilization.

DDL Analyzer - the Data Description Language Analyzer that transforms a DDL schema specification into an internal data dictionary and initializes a corresponding ENA data base.

DML - the complete Extended Network ArchitectureData Manipulation Language whose commands can be embedded in a wide variety of host languages.

DMU - the Design Modification Utility that a data base administrator uses to monitor space utilization, expand data base size, and modify user access privileges.

GURU - the expert system development tool and environment.

IDML - the Interactive Data Manipulation Language for interactive DML training, testing DML logic, and ad hoc data base surgery.

KnowledgeMan/2 - the decision support environment with a complete set of synergistically integrated knowledge management facilities including spreadsheet, graphics, text processing, procedural modeling, local data management, SQL inquiry, etc.; also a valuable prototyping tool for developers.

PVRU - the Physical Validation/Repair Utility can be used to check and, if needed, repair a data base.

QRS - the non-procedural, English-like Query Retrieval System for satisfying ad hoc and "what-if" information needs.

RCV - the Recovery Utility that a data base administrator uses to monitor the transaction log, roll the data base back to a former state, and selectively recover from data base corruption.

RDL Analyzer - the Report Definition Language Analyzer that a developer uses to automatically generate interactive C application programs containing all necessary I/O statements, computations, and DML logic.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

MANUAL ORGANIZATION

Chapter I of this manual provides an introduction to the family of MDBS III application development tools. This chapter illustrates how MDBS software modules interact with MDBS III data bases and how the software modules complement each other. This chapter concludes with a summary of the DDL features and associated software.

Chapter II discusses the MDBS constructs for building a schema (data base structure). Data item definitions, data record definitions, inter-record relationships (that is, set relationships), and data base area structures are included here. This chapter also contains an example schema.

Chapter III documents the Data Description Language syntax. Each section of the DDL source specification appears with all optional clauses. Also, a sample DDL source specification of the Chapter II schema is included in Chapter III.

Chapter IV discusses data security specifications. Items such as passwords for legitimate data base users, data encryption for sensitive data fields, and user access codes for restricting data usage are presented in this chapter.

Chapter V discusses automatic data integrity facilities. Topics of this chapter include specific data ranges for any data type and limits to prevent set relationship changes between any two data base records.

Chapter VI describes the DDL Analyzer software. Called MDBS.DDL, this software includes a small text editor to create or modify a DDL source specification. Also, this chapter discusses the interactive uses of the DDL Analyzer.

Chapter VII discusses the diagnostics that can arise during DDL analysis. Each diagnostic message includes a brief problem synopsis and one or more possible causes of the problem.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

DATA BASE DESIGN

An initial step in developing an application system is designing the logical structure (schema) of the data base. The schema shows the kinds of data and the data relationships that will exist in the data base.

The expressive power and flexibility that MDBS III provides for schema designs that can directly represent every kind of naturally occurring relationship commonly encountered in real world applications. With MDBS III, the schema conforms to the application. You can design schemas that accurately reflect real world data relationships. Every relationship in the schema is unique and MDBS maintains the integrity of each relationship.

MDBS directly supports three fundamental relationships which can exist between data records:

- one-to-one relationship - allows one record occurrence as a set member for a given record occurrence established as a set owner; also, each record established as a set member can have at most one record occurrence as the set owner.

- one-to-many relationship - allows several record occurrences as a set member for a given record occurrence established as a set owner; also, each record established as a set member can have at most one record occurrence as the set owner

- many-to-many relationship - allows several record occurrence as a set member for a given record occurrence established as a set owner; also, each record established as a set member can have several record occurrences of the set owner.

Refer to Chapter II for definitions and examples.

MDBS supports two variants of fundamental set relationships. The first variant is recursive relationships. A recursive set relationship exists when a single record type is defined as both the set owner and set member (Figure I-3).

The second set variant MDBS supports is forked relationships. A forked set relationship exists when multiple record types are defined as the set owner or set member (Figure I-4). A doubly forked set (which has multiple owner record types and multiple member record types) is also supported by MDBS.

Multiple relationships can exist between the same pair of record types and remain distinct. There is no restriction on the overall configuration of the relationships in a schema.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

Each MDBS schema accurately models the environment it represents, with every relationship depicted in a clear, self-documenting way.

Each employee has at most one biography and each biography belongs to at most one employee.

Each customer can place many orders and each order is placed by at most one customer.

Each supplier can supply many parts and each part can be supplied by many suppliers.

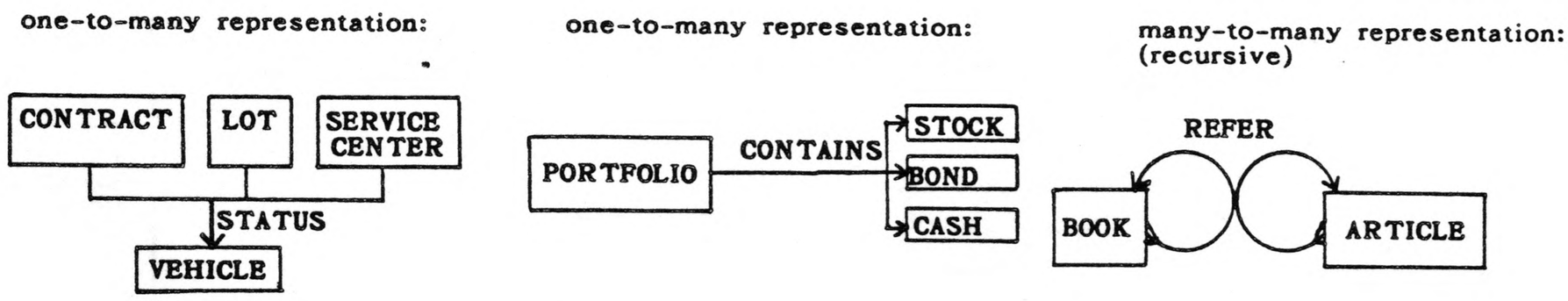
Figure I-2. Examples of the Three Kinds of Relationships between a Pair of Record Types

In a critical path, each event follows at most one other event and is followed by at most one other event.

Each employee can manage many other employees and is managed by at most one other employee.

Each action can be the prerequisite for many other actions and can have many other actions as its prerequisites.

Figure I-3. Examples of the Three Kinds of Recursive Relationships Involving a Single Record Type



Each contract, rental lot, or service center can involve many vehicles and each vehicle can be either under contract, in a lot, or in a service center.

Each portfolio can contain many positions in stocks, bonds and/or cash. Each stock, bond or cash position is contained in at most one portfolio.

Each book and/or article can refer to many other books and/or articles and can be referred to by many other books and/or articles.

Figure I-4. Examples of Some Forked Relationships

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

DDL Analyzer

The MDBS III Data Description Language is a free-form language to formally specify the schematic structure of a postrelational data base. It supports nine data item (field) types and numerous set relationship orderings such as last-in-first-out (lifo), first-in-first-out (fifo), next, prior, and sorted (single or multiple fields). Built-in sorting sequences are provided for several languages including English, Danish, French, German, Swedish, Norwegian and Finnish. In addition to specifying a data base's logical structure, the DDL can control physical placement of data records, prescribe data security conditions, and define data integrity constraints.

The DDL Analyzer (MDBS.DDL) provides an environment to develop schema specifications and to initialize new data bases, as illustrated in Figure I-5. In batch mode, MDBS.DDL can analyze a schema specification and initialize a data base. In interactive mode, MDBS.DDL provides the following functions:

- create, save, and retrieve the DDL specification of the schema for a given application
- analyze the DDL source specification for consistency
- Issue error messages for DDL source specification inconsistencies
- Edit DDL source specifications
- generate a data dictionary from the DDL source specification and initialize a corresponding data base.

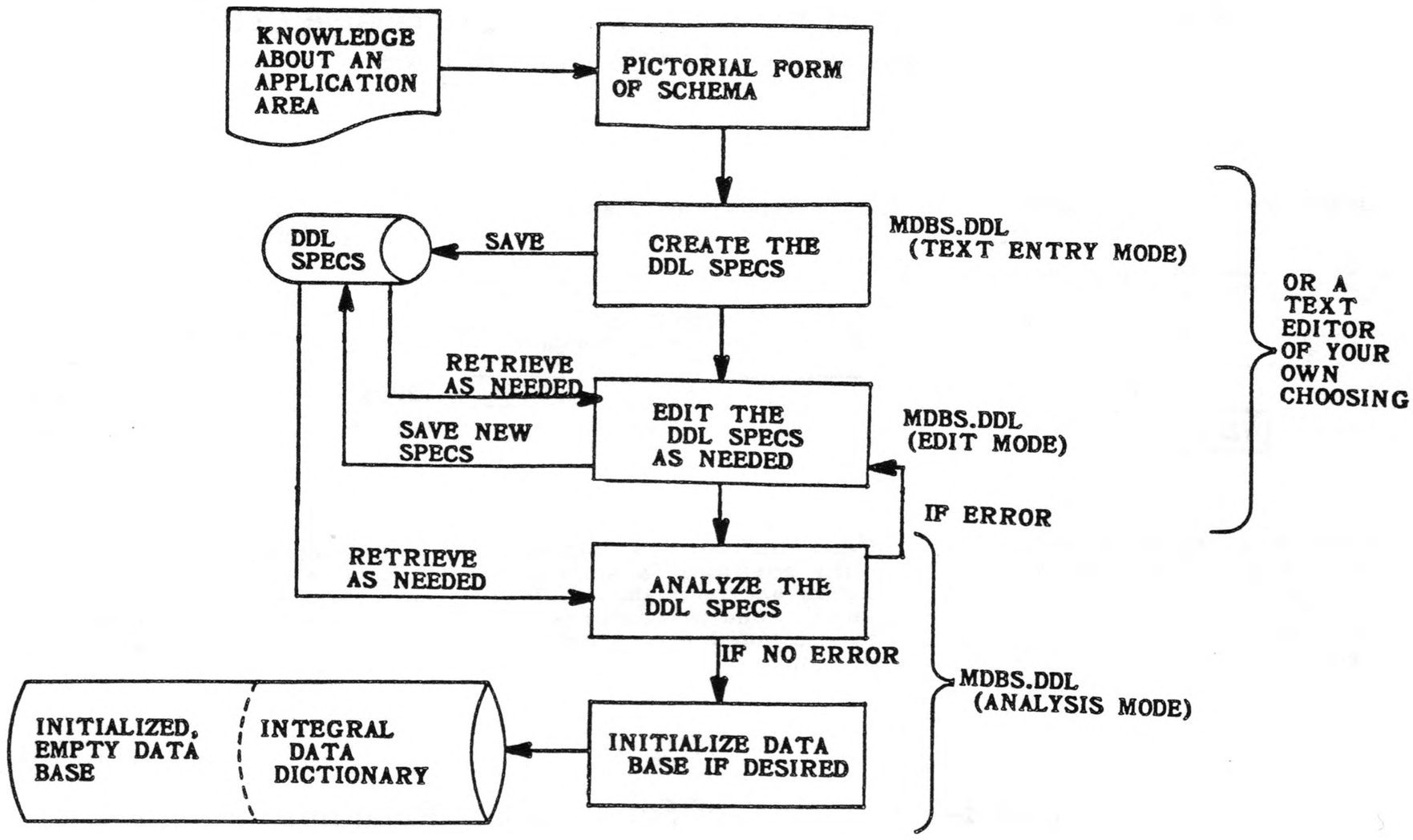


Figure I-5. Using the MDBS.DDL Module

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

Chapters II - VII of this manual describe the MDBS Data Description Language and DDL Analyzer operations in detail.

DBRS

The MDBS III Data Base Restructuring System (DBRS) supports all types of schematic restructuring for existing MDBS III data bases. This includes adding and removing data fields, record types, or relationships for MDBS schemas. Data Base contents are automatically revised to conform with the new schematic structure. This optional software module is documented in the MDBS Data Base Restructuring System Manual.

ACCESS LANGUAGES

MDBS III supports a variety of methods to access MDBS data bases. Each access method is called an access language and fills a particular processing need or purpose. Each access language relies only on the schema definition stored in the data dictionary of the data base. Knowledge of the physical organization of the data within the data base is unnecessary. Processing with an access language can begin anywhere in a schema and any schematic relationship can be processed in any direction.

The ability of MDBS III access languages to support refined field-oriented processing results in a high degree of data independence. This attribute insulates application programs and queries from schematic changes.

The access languages use the Data Base Control System (MDBS.DMS) to process all data requests. This software module guards data security and integrity, monitors concurrent processing situations, manages the virtual memory, and controls all internal physical aspects of data access. The data base control system is independent of host programming languages.

The data base control system (MDBS.DMS) is a collection of data base management functions. When an access language calls one of these functions, the data base control system performs all the physical details required to perform the requested function. The data base control system consults the data dictionary in the course of performing its functions.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

DML

The MDBS III Data Manipulation Language (DML) provides a complete set of commands to navigate, store, retrieve, and modify data within an MDBS data base. Your application program contains DML commands that interact with the data base control system. When an application program performs a DML command, it makes a request to the data base control system. The data base control system performs the command and returns control to the calling program.

The DML commands include functions to process a record at a time. There is also a group of extremely powerful Boolean DML commands, each of which locates zero, one, or more records that satisfy specific criteria. The MDBS III DML allows many records to be simultaneously available for immediate, efficient processing.

MDBS III DML commands can be invoked from application programs written in COBOL, Pascal, BASIC, FORTRAN, C and assembler. You can also directly invoke DML commands as innate operations at any point within the KnowledgeMan/2 environment (an interactive decision support environment) or within GURU (an expert system development tool and environment). You can also embed DML commands in programs written in the KnowledgeMan/2 programming language or in GURU expert system rule sets. See the MDBS Data Manipulation Language Manual for more information on the DML commands.

The working relationship between an application program and MDBS.DMS in an application system is shown in Figure I-6.

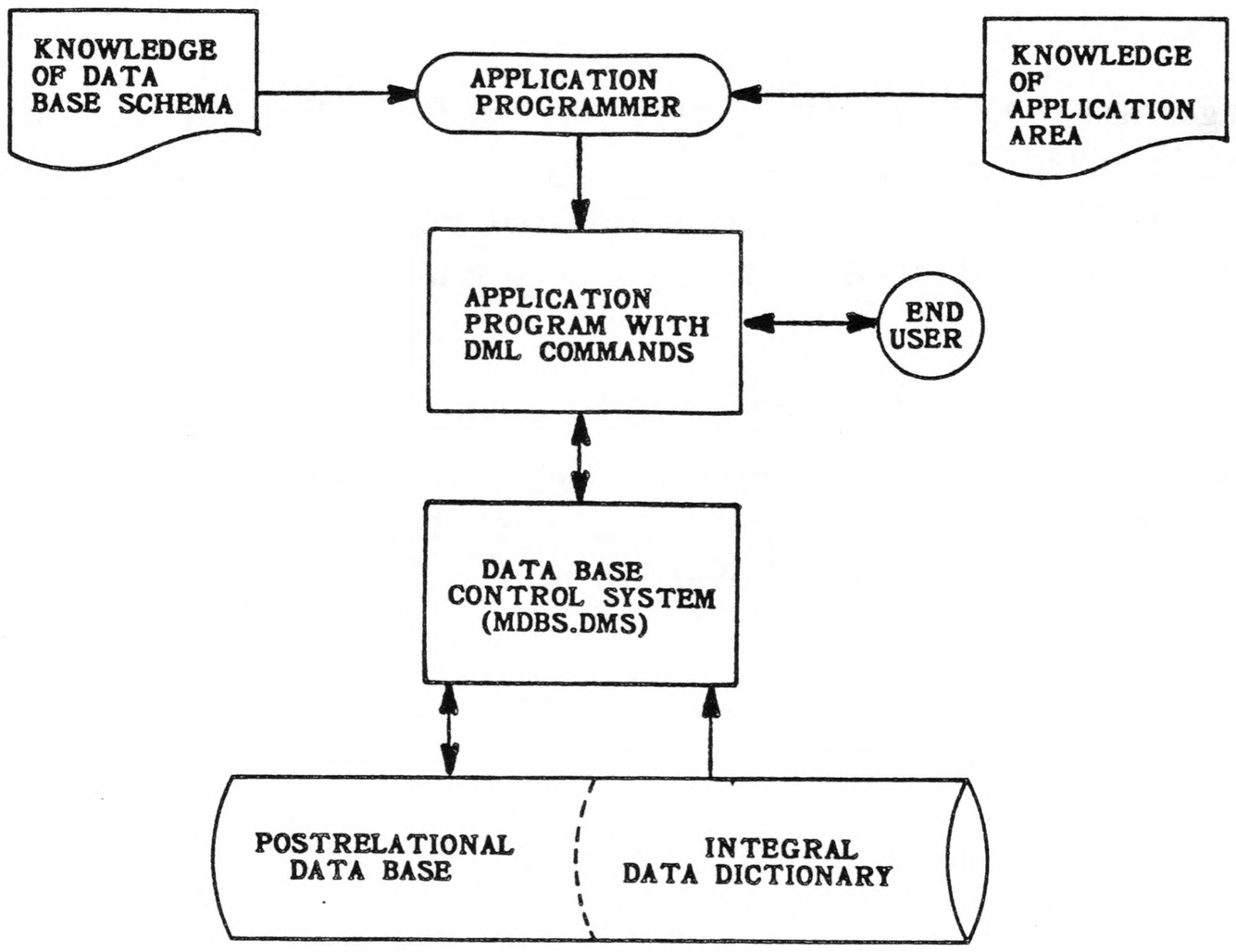


Figure I-6. Using the MDBS.DMS Module

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

MDBS DML provides a standard, uniform language for handling all data manipulation regardless of the application system being developed and regardless of:

- the central processor being used
- the mixture of disk drives available
- the operating system being employed
- the host languages desired

IDML

The MDBS III Interactive Data Manipulation Language (IDML) is an interactive environment that supports most DML commands. It is independent of all programming languages. This optional access language is especially useful for DML training, testing DML logic, and performing ad hoc data base changes. The IDML software module includes the following functions:

- built-in help facility
- DML navigation commands
- DML storage and retrieval commands
- DML editing commands
- schema display commands (data dictionary inquiries)
- parameterized macros for frequently used command sequences

See the MDBS Interactive Data Manipulation Language Manual for more information on the IDML environment.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

QRS

The MDBS III Query Retrieval System (QRS) accepts nonprocedural structured English requests from non-programmers. After analyzing a query request, QRS internally employs the necessary DML logic for accessing desired data from a data base. This data is then presented as a report in tabular form or saved as a flat file on disk. QRS can be used directly from the operating system or it can be used at any juncture within an interactive KnowledgeMan/2 or GURU session. In either mode of operation, QRS is particularly valuable for satisfying the ad hoc information needs of end users, developers and administrators.

Included among the many QRS features are user-definable words and commands, tabular end user views, virtual fields, data dictionary viewing, environment control options, fully selective retrieval, arithmetic expressions, dynamic output sorting, control breaks (with full statistics), value labels, batched query execution, and a Data Interchange File (DIF) output capability. All aspects of the query language and the optional QRS module are presented in the MDBS Query Retrieval System Manual.

RDL

The MDBS III Report Definition Language (RDL) is a report program generator. RDL uses a nonprocedural language to formally specify the characteristics of a desired report and the prompting behavior of a program that will generate the report. From an RDL input specification, the RDL Analyzer produces a C language program containing everything (all DML logic, computations, input prompts, and output formats) needed to generate a report having the desired content and appearance.

RDL features include extensive output controls, built-in statistical functions, user-definable functions, free-form layouts for report details, detail sorting, headers and footers for pages and detail groupings, multi-level grouping, fully selective retrieval criteria, wildcard string and character matching, character class matching, and computed variables. The MDBS Report Definition Language Manual contains a full description of the Report Definition Language and the RDL Analyzer.

BLF

The MDBS III Batch Load Facility (BLF) is an optional module that automatically loads the contents of external flat files into an MDBS data base. From a single file it is able to create data base records of various types and correctly establish their inter-relationships within a data base. Thus, BLF allows data from external sources to be incorporated into a data base without any programming effort. This software module is fully described in the MDBS Batch Load Facility Manual.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

DATA INTEGRITY FACILITIES

Preserving an application system's data integrity is essential. MDBS III provides a variety of automated facilities for reducing the possibility of integrity breaches and for recovering from any data base destruction or corruption that may occur. These integrity facilities exist above and beyond MDBS III's extensive data security mechanisms for preventing unauthorized modifications to the data base.

The MDBS III data base control system automatically prevents a data value from being stored for a field if it is not consistent with the field's type. The data base control system also enforces the data base integrity in the following ways:

- valid data values of all data items
- feasibility ranges for all data items
- all one-to-one and one-to-many relationships
- fixed relationship retention

To reduce the possibility of data base corruption due to external factors, MDBS offers the following items:

- developer controlled buffer flushing
- page image posting - allows warm restarts in single user processing situations (available with RTL data base control system)
- transaction logging with selective recovery (available with RTL data base control system)
- Physical Validation/Repair Utility (PVRU)

Transaction logging is pertinent to both single user and multiuser applications. The RTL data base control system can automatically log all transactions to a log file. The scope of logical work units is developer controllable. Each transaction is accompanied by the identity of the user who performed it. In the event of data base corruption, the Recovery (RCV) Utility can regenerate a valid data base up to the point of the last completed logical work unit. Selective regeneration is also permitted to allow data base contents to be rolled back to any desired earlier state. The RTL form of MDBS is fully described in the MDBS Recovery and Transaction Logging Manual.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

The Physical Validation/Repair Utility detects damage within a data base and provides repair functions to a data management system. By examining the internal physical structure of a data base, PVRU can be used to routinely check a data base for early signs of damage and ease data maintenance costs. The PVRU utility is fully described in the MDBS Physical Validation/Repair Utility Manual.

DATA SECURITY FACILITIES

The security mechanisms built into MDBS III are extensive, even by mainframe standards. Security control information for a data base is maintained in the integral data dictionary. This includes a list of valid data base users, a password and access privileges for each user, plus the access restrictions defined for each field, record type, and relationship existing in the schema. The security control information itself can be viewed and modified only by authorized users of DBRS or the MDBS III Design Modification Utility (DMU). Design modification capabilities of the optional DMU module are described in the MDBS Design Modification Utility Manual.

Data Base Users

To be able to use any of the MDBS III access languages or utilities for data base access, a bona fide user name and associated password must be provided. The retrieval and/or modification access privileges of that user are automatically observed by the data base control system. If a user's privileges are not a subset of the access restrictions on a data resource, then that user's attempts to access it are denied. One of the more innovative aspects of this security mechanism is the ability to define access restrictions for data relationships.

Data Encryption

MDBS III supports fully selective encryption, allowing the developer to control which fields are to have their values automatically encrypted. This encryption occurs for data held in both the data base and any of its transaction logs.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

PERFORMANCE TUNING FACILITIES

MDBS III possesses several design features that provide fast, efficient processing. Beyond these default performance benefits, MDBS III provides a variety of performance tuning controls that allow the developer to adjust certain aspects of the behavior of the data base control system to maximize its performance with respect to a particular application system. The tuning facilities for the data base control system include the following items:

- automatic data compression for designated fields
- assignment of record types to desired areas (that is, devices)
- control over area page sizes
- physical clustering of logically related records
- variable field replication
- record storage and retrieval via hashing
- multiple secondary key indexes for any record type

In addition, a developer can use the MDBS III Data Base Restructuring System (DBRS) to dump and rebuild an existing data base so the physical placement of the data base contents are optimized with respect to data base design criteria.

MULTIUSER PROCESSING

Ensuring the integrity of a data base that is being used concurrently by multiple users is a crucial concern for application system developers. Versions of MDBS III implemented for multiuser environments provide the developer with a complete range of locking and contention controls. These enable the developer to effectively avoid disastrous consequences that can result when one end user attempts to either view or alter data that another user is updating.

For maximum flexibility and performance optimization, locking can be applied to individual records, relationships, or groups of records. MDBS III supports both passive and active locking techniques. A contention count protocol is provided to release the system from potential deadlock situations. For each application program, the developer can control the number of automatic retries that will occur whenever an attempt is made to access a locked resource. The time interval between retries is also developer controllable.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

The MDBS III locking and contention controls are the same across all supported multiuser operating systems and local area networks. These include the UNIX, XENIX and ULTRIX multiuser operating systems and the IBM, Novell, and 3COM local area networks. The underlying data base control system implementation is dependent on the nature of the multiuser environment. Characteristics of MDBS III multiuser processing that are applicable to particular environments are documented in pertinent **MDBS System Specific Manuals** and **Local Area Network Supplements**.

MDBS III AND KNOWLEDGEMAN/2: DECISION SUPPORT

KnowledgeMan/2 is the highly acclaimed decision support environment that provides end users with a means for satisfying many of their own knowledge management needs. By helping end users and casual developers to be more productive, KnowledgeMan/2 frees the professional application developer to concentrate on truly challenging applications involving large data bases. KnowledgeMan/2 can directly access these massive MDBS III data bases subject to all of the usual security and integrity controls enforced by the MDBS III data base control system. The tight integration of MDBS III with KnowledgeMan/2 makes the data held in massive MDBS III data bases susceptible to fast local decision support processing, without endangering the security or integrity of these central data bases.

KnowledgeMan/2 encompasses a broad spectrum of knowledge management facilities for supporting decision makers. These facilities (fully described in the KnowledgeMan/2 Reference Manual) include the following functions:

- relational data base management
- spreadsheet analysis
- business graphics
- full-screen text processing
- customized report generation
- ad hoc, SQL inquiry
- mathematical functions and calculations
- forms management
- customized procedures
- remote communications

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

KnowledgeMan/2's facilities are not merely separate tools operating within a window manager. KnowledgeMan/2 eliminates the cumbersome process of "cutting and pasting" and the inconvenience of being forced to switch back and forth among different tools and windows. With KnowledgeMan/2, a user is not forced to carry on all processing within the constraints of a single dominant component such as a spreadsheet or outline processor.

KnowledgeMan/2 is based on an integration philosophy that allows any facility to be used independently of the others. It allows multiple facilities to be used together within a single operation, resulting in a total effect that is much greater than the sum of individual effects.

KnowledgeMan/2 has been carefully designed to give users plenty of room to grow. It is simple enough that novice users can become productive in a very short time. As experience and needs increase, a user is free to grow in any desired direction with KnowledgeMan/2's universe of knowledge management possibilities. One direction is toward the use of MDBS III data bases. For instance, in the midst of a KnowledgeMan/2 session QRS can be invoked to query a postrelational data base. Upon exiting QRS, KnowledgeMan/2 processing resumes and a single request will incorporate query results into KnowledgeMan/2's local data base. Furthermore, MDBS III application programs for processing data bases can be executed at any desired juncture within a KnowledgeMan/2 session. These programs can use data exported by KnowledgeMan/2 for updating a postrelational data base or they can extract information from the data base for subsequent import into KnowledgeMan/2's local processing environment.

There is an even closer connection between MDBS III and KnowledgeMan/2. As explained in the KnowledgeMan/2 Reference Manual, **the entire MDBS III data manipulation language is available as a fully integral component of KnowledgeMan/2.** Any DML command can be invoked at any point in an interactive KnowledgeMan/2 session to transfer data between a central postrelational data base and local KnowledgeMan/2 variables, arrays, spreadsheets, or relational tables. DML commands can also be embedded in the cell definitions of spreadsheets and in customized KnowledgeMan/2 procedures. Application developers can use these latter two techniques to provide KnowledgeMan/2 users with direct access to the latest information held in a central MDBS III data base. This requires no DML comprehension on the part of KnowledgeMan/2's end users.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

MDBS III AND GURU: EXPERT SYSTEMS

GURU is an expert system development tool. In addition to its extensive facilities for business computing (spreadsheet, procedural modeling, graphics, etc.), GURU provides full-scale facilities for expert system construction and expert system consultation. All of these are blended together into a single environment. GURU is well-suited for developing application systems with their own built-in expert reasoning capabilities. Expert systems built with GURU can directly process spreadsheets, relational data tables, models, forms, etc. In addition, they can directly access MDBS III data bases subject to all the usual security and integrity controls enforced by the MDBS data base control system.

Ordinary expert system development tools do not support even the most basic kinds of knowledge representation and processing: real data base management, spreadsheets, structured programming, business graphics, remote communications, forms management, text processing and so on. An expert system is constructed by building a set of rules that capture an expert's knowledge about solving problems in an application area. Unlike ordinary tools, GURU allows rules to be specified in terms of spreadsheet calculations, relational data base retrieval, statistics generation, graphics display, model execution, MDBS DML commands, and so forth.

When an end user consults an expert system, GURU's innate inference engine reasons with the rule set in order to reach a solution. End users can directly ask for a consultation or the developer can embed consultation requests in application programs that end users execute. In either case, the inference engine is able to carry out all business computing actions specified in the rule set's rules. GURU's inference engine is able to use both forward and reverse reasoning, accommodate fuzzy variables, reason about uncertain situations, and explain the line of reasoning it used in reaching a solution. GURU's capabilities are fully described in the two-volume GURU Reference Manual.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

MDBS DATA DESCRIPTION LANGUAGE AND SOFTWARE SUMMARY

The remainder of this manual concentrates on the Data Description Language that can be used to formally specify schemas designed for a postrelational data base. It also focuses on the DDL Analyzer which is the software module provided for processing a DDL specification. This chapter concludes with summaries of the DDL and DDL Analyzer features.

Summary of MDBS DDL Data Structuring Features:

- Up to 255 record types are allowed per logical data structure.
- Variable and fixed length records are supported.
- Up to 65535 data items (fields) are allowed per record type.
- The following kinds of data items are allowed: character, integer, real (floating point), string (variable length character data item), binary (variable length data item), unsigned, internal decimal, time, date.
- Encryption of a data item's values is supported.
- A feasibility range can be specified for a data item's values.
- The following kinds of sets are allowed for defining relationships among record types: one-to-many and many-to-one (i.e., CODASYL set), one-to-one, many-to-many.
- A record type can be the owner of many sets (of any kind) and the member of many sets (of any kind); many sets (of any kind) can exist between two record types.
- Any set can be forked into multiple member record types and/or multiple owner record types.
- Recursive sets are permitted.
- Sorted, LIFO, FIFO, next, prior, and immaterial orderings supported for member records of a one-to-many set. These orderings are supported for both owner records and member records of a many-to-many set.
- A sorted ordering can be ascending or descending and can be based on multiple data item types.
- Sorted orderings can be based on English sequences or on any of a variety of non-English (e.g., French) sequences.
- Automatic or manual record insertion for sets.
- Fixed or optional record retention for sets.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Overview

- Aside from the main data base area, up to 15 extra (physically distinct) areas can be defined for holding data base records. In some environments, there is a maximum of seven extra areas.

- Occurrences of a record type can be assigned to one or more areas.

- Related occurrences of more than one record type can be physically clustered.

- Records can be "hashed" into the data base for fast, direct access.

- An extensive access code mechanism is provided for automatic enforcement of read and write access constraints placed on each data base user.

- Record interrelationships are not implemented by redundancy of data values (as in flat file systems), nor are they implemented by pointer chaining (as in most CODASYL network and hierarchical systems). For superior performance, MDBS record interrelationships are implemented with a proprietary technique involving enhanced, multi-level, indexed dynamic pointer arrays.

Summary of the DDL Analyzer (MDBS.DDL) Features:

- Supports free format DDL source text.

- Can be used in batch manner to analyze a DDL source text file and initialize a data base for it.

- Can be used interactively in text entry mode which supports:

  - Creating DDL source text
  - Listing DDL source text
  - Deleting DDL source text
  - Saving DDL source text on a disk file
  - Retrieving DDL source text from a disk file
  - Renumbering DDL source text

- Can be used interactively in an edit mode which allows line editing of DDL source text.

- Can be used interactively in an analysis mode which:

  - Analyzes DDL source text for consistency
  - Displays error message if error detected
  - Can initialize the data base if no error is detected

© Copyright 1987 Micro Data Base Systems, Inc.

II. DEFINITIONS AND EXAMPLES

DATA BASE AREAS AND PAGES

A **data base** consists of one or more areas. An **area** exists on a disk file. Each area consists of two or more pages. A **page** is a fixed-size block of bytes or words that is transferred as a logical whole between central memory and disk. This transfer is automatically performed as needed by MDBS.DMS in order to carry out DML commands. Each area can hold up to 4095 pages (8191 in some environments). A few pages in each area are reserved for use by MDBS.DMS. These are called System pages. System pages contain data dictionary (and other) information. A DML command to store data will cause the MDBS.DMS to place that data on the non-System pages in an area.

The name of a data base is stated with the DDL. When the data base is initialized, one area having the same name as the data base is automatically prepared for use. This area is referred to as the **main** data base area. The System pages of this area contain the data dictionary. Additional areas (up to 7 or 15 more) can be optionally defined with the DDL. The DDL is used to give each of these extra areas a name and each is automatically prepared for use when the data base is initialized. The designer can control:

- the name of the file on which an area resides
- the disk on which an area resides
- the disk drive to which an area is assigned

This means that all data base areas do **not** need to be on-line when the data base is initialized by MDBS.DDL or when the data base is used by MDBS.DMS in response to a DML command (or query). However, the area having the data base name (i.e., the main area) should always be kept on-line, since it contains the data dictionary. MDBS.DMS needs to consult the data dictionary to carry out DML commands. Part of the data base can exist on fast drives, while another part of the data base is on relatively slower (less expensive) drives.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

For example, the following data base spans multiple disks, disk drives, and files. This data base is presented in table II-1 and diagramed in Figure II-1.

Table II—1

Data Base Area Name	System Pages	Total Pages	File Name	Hard Disk	Disk Drive
DBEX	3	18	FILEA	DISK1	DRIVE1
AR4	2	100	FILEB	DISK2	DRIVE2
BAREA	2	165	FILEC	DISK2	DRIVE2
AR3	2	50	FILED	DISK3	DRIVE2

Suppose an application program needs to access data from the DBEX and BAREA areas. Disk1 is mounted in Drive1 and Disk2 is mounted in Drive2. Suppose that another application program needs to access data from the DBEX, AR3, and BAREA areas. Since both AR3 and BAREA are assigned to the same drive but are on different disks, they cannot be used simultaneously. However, the application program can prompt the end user to assure that the appropriate disk is in Drive2 at the proper time. Simultaneous usage of the two areas is accomplished by allowing one of them to reside on Disk1 or by allowing them to reside on the same disk or by assigning one of them to a third drive. The file name and drive for an area can be altered with the DML when that area is "opened" in an application program.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

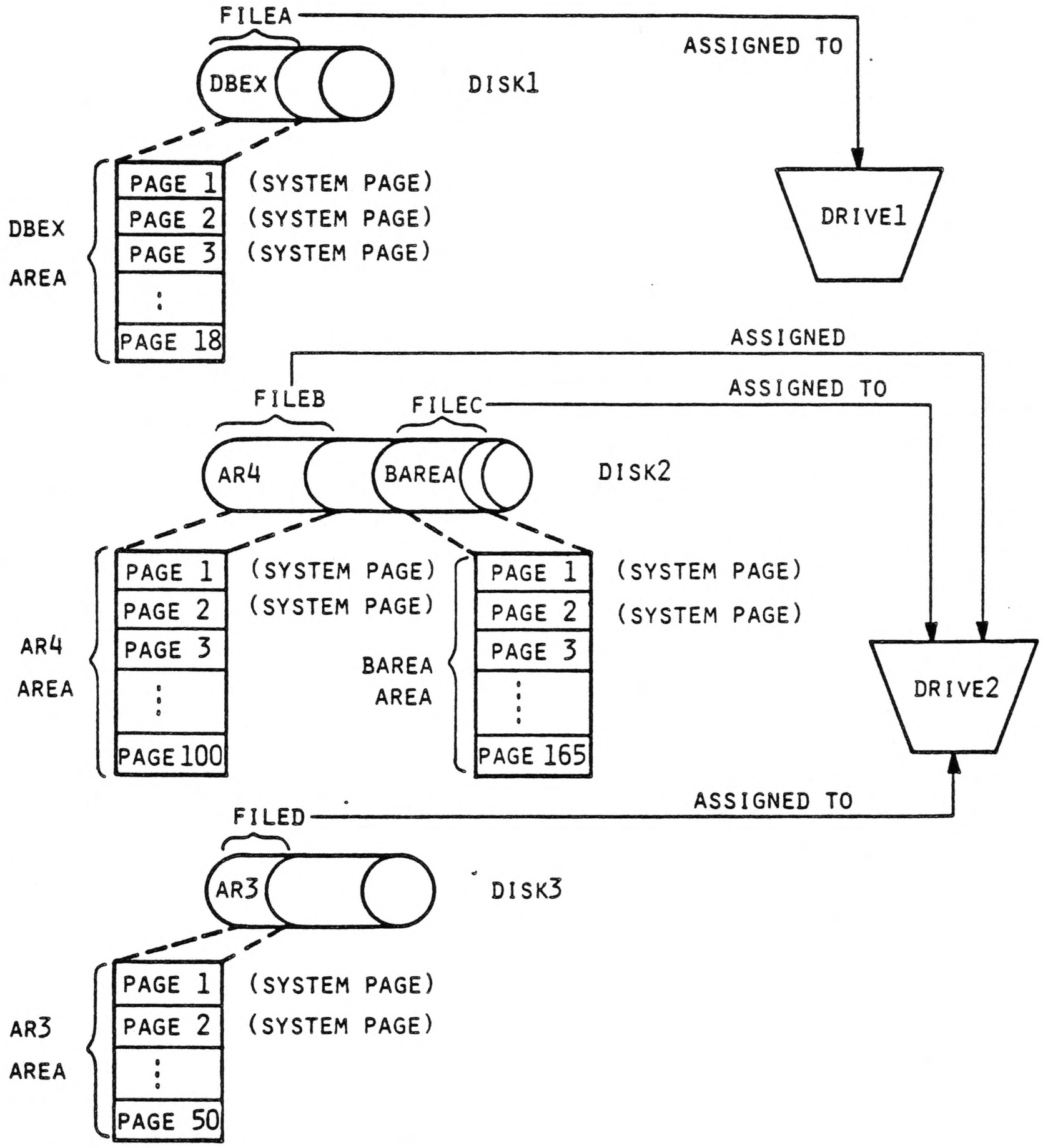


Figure II-1. Areas and Pages

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

The selection of an appropriate page size is important, since it influences the number of pages in memory during system execution and thus impacts the performance of the system. If the memory available for page buffers is relatively limited, it is advisable to select a small page size to maximize the number of pages in memory. For instance, performance with four small pages in memory will typically surpass performances based on two larger pages in memory. On the other hand, if the buffer region is large, then a comparatively large page size can be advantageous, providing a minimum of about 5 to 8 of these pages will fit in the buffer region.

There are certain lower limits on the page size. For Version 3c, it must be at least 256. For Version 3a, the page size must be a multiple of 256 and, furthermore, it must be large enough to hold an occurrence of the largest record type in the area. Another consideration involved is how the record occurrences are distributed on a page. It is possible (depending on the size of the record) that records are put on a page in such a way that a great deal of room is "wasted" on each page. As an example, consider the extreme case where **only** one record type of 260 bytes has been assigned to an area with a page size of 512 bytes. Since records are not split over two or more pages, this would mean that only one record is being stored on a page. This can result in a great deal of "wasted" space, except that MDBS will attempt to use such space to hold internally used indices. Consideration of factors such as these aid the application developer in selecting a good page size for an area.

THE LOGICAL STRUCTURE OF AN MDBS DATA BASE

The **logical structure** of a data base is specified in terms of data items, record types, and sets. A **record type** is a named group of zero, one, or more data items. **Data items** (fields) are the most elemental components of a logical structure. Every data item is given a name and is described in terms of the type and size of its values. The permissible types and sizes are discussed later in Section II-C. Two data items in the same record type cannot be given the same name (the same item name may be used in different record types).

As an example, we may want a data base to contain information about all departments in an organization. For each department we may want to keep track of the department's number, name, and location. The logical structure for such a data base would have a record type (call it DEPT) which consists of three data items: call them DNUMBER, NAME, LOCATION. Pictorially, a record type is represented by a labeled rectangle, with the names of its data items inside. The DEPT record type is represented as shown in Figure II-2. This Figure also shows four other record types: EMPLOYEE, BIOGRAPH, JOB, SKILL. No two record types can have the same name. The meaning of the arrows in this logical structure is discussed later.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

For each record type declared in a data base's logical structure, there can exist zero, one, or more occurrences of that record type in the data base itself. An occurrence of a record type consists of a value for each of the record type's data items. For instance, an occurrence of the DEPT record type might be [101,SALES,CHICAGO]. Another occurrence of the same record type might be [108,FOUNDRY,DETROIT]. A sample occurrence of the record type JOB is [A15,JANITOR]. MDBS places no upper limit on the number of occurrences for a record type that can exist in a data base.

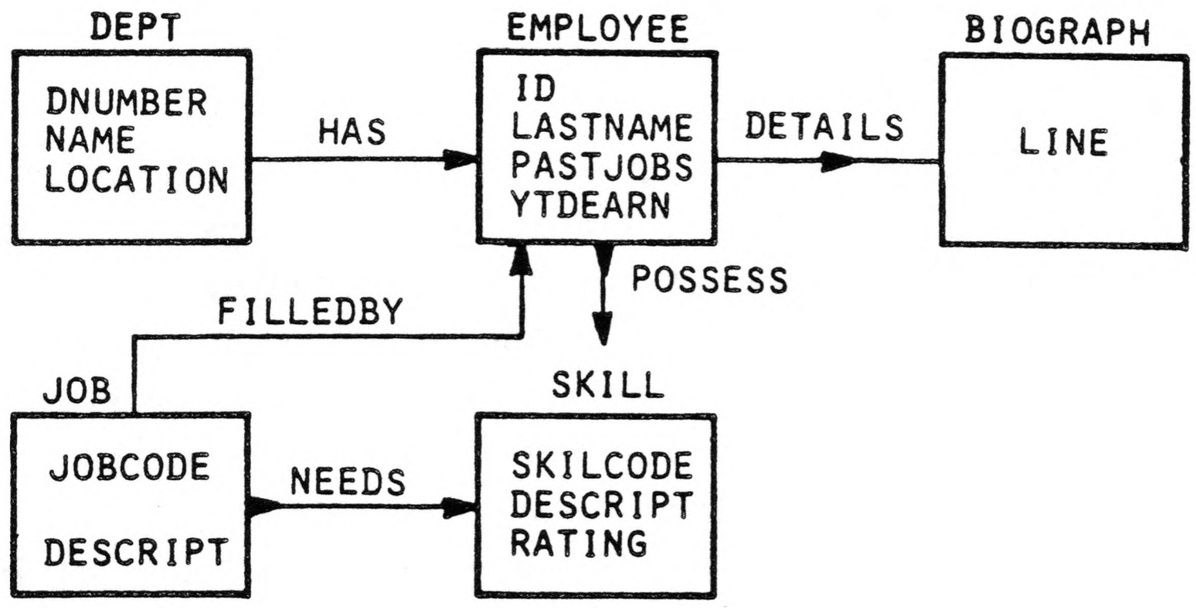


Figure II-2. A Sample Logical Structure

A data item can optionally be specified to be an array. It is then called a repeating item. For instance, PASTJOBS in EMPLOYEE could be a repeating item of length 3, if we desire to keep track of the three most recent jobs of each employee. A data value of PASTJOBS would consist of an array of length 3 (i.e., an array having three entries).

An occurrence of a record type is called a **record occurrence** (or simply: a record). A value for a data item is called an **item occurrence**. Thus a record occurrence exists in the data base and contains an item occurrence for each data item in the record type. It must be emphasized that **item occurrences and record occurrences exist in the data base** itself, while **data items and record types** are used to **define the logical structure of the data base**. The DDL is used to specify a data base's logical structure in terms of data items and record types. The DML is used to create, modify, and extract item occurrences and record occurrences of a data base. This data manipulation is always made in accordance with the data base's logical structure. To create a new occurrence of the DEPT record type, a DML create command is qualified by DEPT. If another application program needs to generate a list of all employee names, it would use the DML to request all record occurrences of EMPLOYEE and for each of these it would request the values of the LASTNAME and FNAME data items. Needless to say, MDBS.DMS automatically performs the tasks of physically locating the record occurrences, bringing them into central memory, and depositing the desired data values into application program variables.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

As each record type is defined in the DDL, the designer has the option of assigning it to one or more areas. This means that when MDBS.DMS creates occurrences of that record type, they are stored in the assigned area(s). More than one record type can be assigned to the same area. One result is that frequently accessed record occurrences can be clustered for use on a fast disk drive. Another result is that record occurrences with particularly sensitive information can be physically protected, by assigning them to an area on a separate disk. This disk is manually placed on-line only for users with sufficient security clearance.

Relationships Between Record Types

In designing the logical structure of a data base, it is vital for an application developer to have some means for defining relationships among record types. In systems that support only flat file structures, this is accomplished by redundancy. These flat file structures are also called "relational" or tabular structures. One or more data items in one record type are repeated in another record type. Occurrences of one record type that are related to a given occurrence of another record type can then be found by looking for matching values between the repeated data items. For instance, to represent the fact that a department has employees, the DNUMBER data item of DEPT could be repeated as a data item in EMPLOYEE. For a given occurrence of EMPLOYEE, a DEPT occurrence with a matching value for DNUMBER could be found. Conversely, for a given occurrence of DEPT, all EMPLOYEE occurrences having a matching DNUMBER value could be found. This approach to representing a relationship between two record types (cross-referencing by redundancy) is entirely permissible within MDBS data structures.

However, it must be pointed out that the flat file approach to data structuring has a number of inherent drawbacks. It does not convey the semantics of a relationship. This is important when multiple, semantically-distinct relationships exist between two record types. From a data integrity standpoint, there is no structural mechanism for restricting a relationship to be a one-to-many or one- to-one relationship. In looking at a flat file structure, the relationships between record types are not always easy to pick out, especially if two record types are indirectly related through several other record types. This problem is compounded for realistic logical structures, which typically involve many record types (not just two or three). The number of record types in a flat file structure proliferates rapidly if the application developer decides to use a third normal form (from relational theory). For realistic application development, flat file structures tend to cause greater storage and processing overhead than do the alternative approaches to representing inter-record type relationships (discussed below).

Although the application developer using MDBS is free to define flat file structures, he should be aware of the inherent limitations and inconveniences of the flat file approach. The application developer using MDBS could also restrict himself to hierarchical structures, shallow network structures, or CODASYL network structures. Here again, limitations, inefficiencies, and inconveniences are frequently incurred.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

With MDBS, a named relationship between two record types can be defined as a **set**. One record type is called the **owner** of the set and the other record type is called the **member** of the set.* The term "set" as used in data base management has no relationship to the mathematical notion of a set. In the pictorial representation of a logical structure (e.g., Figure II-2), a set is represented by a line between the owner record type and the member record type. The name of the set is placed alongside the line. An arrowhead appearing on a line always points from the owner record type to the member record type. The positioning of arrowheads on lines is used to distinguish among the various kinds of sets. If an arrowhead is adjacent to the member record type, then there can be many member record occurrences related to each record occurrence; otherwise there is no more than one related member record for any owner record. If an arrowhead is adjacent to the owner record type, then there can be many owner record occurrences related to each member record occurrence; otherwise there is no more than one related owner record for any member record.

Four distinct kinds of sets are allowed in MDBS data structures:

1 : 1 Set for representing a one-to-one relationship
(Owner⟶|member)

1 : N Set for representing a one-to-many relationship
(Owner⟶ member)

N : 1 Set for representing a many-to-one relationship
(Owner⟵| member)

N : M Set for representing a many-to-many relationship
(Owner⟵⟶member)

In all MDBS manuals the term "set", used by itself, refers to all four kinds of MDBS sets.

A 1 : N set (⟶) is a traditional CODASYL set. (It is also analogous to what is called a parent-child relationship in hierarchical systems.) In specifying a 1 : N set, the application developer is stating that one occurrence of the owner record type can be associated with N ($N > 0$) occurrences of the member record type. However, an occurrence of the member record type is associated with **no more than one** occurrence of the owner record type. In Figure II-2, HAS is an example of a 1 : N set. One department can have many employees, but an employee is associated with no more than one department. FILLEDBY is another 1 : N set; a job type can be filled by many employees, but an employee fills only one job at a time. Unlike shallow network systems, in MDBS structures a record type can be the owner of many sets and it can be the member of many sets. Unlike hierarchical systems, a record type can be the member of many sets.

---

* Sets with multiple owner record types and multiple member record types, as well as recursive sets, are also allowed in MDBS; these are explained in a later section entitled "Special Ways of Using Sets".

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

An **N : M set** (►—►) relationship between two record types means that an occurrence of the owner record type may be associated with M (M > 0) occurrences of the member record type. Furthermore, an occurrence of the member record type may be associated with N (N ≥ 0) occurrences of the owner record type. Thus an N : M set is used to **directly** represent a many-to-many relationship between record types. In Figure II-2, NEEDS is an N : M set. A job may need many skills and a skill may be needed by many jobs. POSSESS is another N : M set; an employee may possess many skills and a skill may be possessed by many employees. Such relationships cannot be directly represented in hierarchical, shallow network, or CODASYL network structures. For instance, the CODASYL approach to representing the relationship between JOB and SKILL necessitates the definition of an additional, artificial record type that is the member of a set owned by JOB and the member of another set owned by SKILL. This indirect method for representing many-to-many relationships is supported in MDBS. However, it is recommended that the N : M set be used instead, since it gives a simpler logical structure, makes application programming much easier, requires less storage, and yields faster processing.

A 1 : 1 set (—►—) relationship between two record types indicates that an occurrence of the owner record type is associated with, at most, one occurrence of the member record type and an occurrence of the member record type is associated with, at most, one occurrence of the owner record type. In Figure II - 2, DETAILS is an example of a 1 : 1 set. An employee has no more than one biography and a biography is associated with no more than one employee. LINE is a string (i.e., variable length) data item, which could have a very long occurrence for some employees and a short value for other employees. LINE could have been included in the EMPLOYEE record type. However, if it is less frequently used than other employee data, it is reasonable to place LINE in a separate record type. This means that lengthy biographical data will not be brought into main memory each time an EMPLOYEE record is accessed. It also allows biographical information to be assigned to an area that is usually not on-line and that is utilized by a slow access disk drive.

The **N : 1 set** (►—) is the exact converse of a 1 : N set. For a N : 1 set, a member occurrence can have N (N > 0) owner occurrences. However, an owner occurrence is associated with, at most, one member occurrence.

A **set occurrence** consists of an occurrence of the set's owner record type and all associated member record occurrences (for example, a department and all employees in that department; another example, a job and all skills that job needs). Set occurrences for a 1 : N set do not overlap; they have no record occurrences in common. The same is true for set occurrences of a 1 : 1 set. Set occurrences for a N : M set can overlap. For instance, a SKILL occurrence can belong to more than one JOB occurrence.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

Set Ordering

For a given set, the member record occurrences associated with an owner record occurrence can be declared (in the DDL) to be ordered. This is referred to as the set's **member order**. Similarly, the owner record occurrences associated with a member record can be declared to be ordered. This is referred to as the set's **owner order**. Specifying a member order is meaningful only for 1 : N sets and N : M sets (for 1 : 1 sets and N : 1 sets there is no more than one member record occurrence per owner record occurrence). Specifying an owner order is meaningful only for N : M sets and N : 1 sets (for 1 : 1 sets and 1 : N sets there is at most, one owner record occurrence per member record occurrence). All member records for a given owner record can be accessed in the logical sequence defined by the set's member order. All owner records for a given member record can be accessed in the logical sequence defined by the set's owner order. In either case, the MDBS DML also allows reverse sequence accessing.

There are six allowable set orders : FIFO, LIFO, NEXT, PRIOR, IMMATERIAL, SORTED. These orderings are logical; they do not control the physical placement of record occurrences. Regardless of the ordering chosen, MDBS automatically maintains that logical order during the creation, deletion, and modification of record occurrences. The six kinds of set orders are described with respect to specifying member order. The descriptions would be identical for specifying owner order except that the term "owner" replaces the term "member" and vice versa.

FIFO (first in, first out) - The first member record occurrence that becomes associated with a given owner record is logically the first to be accessed. The second member record to become associated with that owner record is logically the second to be accessed, and so forth.

Example: If the member order for HAS in Figure II-2 is FIFO, then the employees associated with a given department can be accessed on a first in-first out basis.

LIFO (last in, first out) - The last member record occurrence to become associated with a given owner record is logically the first to be accessed. The second-to-last is the second to be accessed, and so forth.

Example: If the member order for HAS is LIFO, then the employees associated with a given department can be accessed on a last in-first out basis.

NEXT - A new member record that becomes associated with a given owner record is inserted into the logical sequence of member records immediately **after** the "current" member record of the set. ("Current" is defined in the MDBS DMS Manual discussion; a DML command allows a user to make any member record "current.")

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

PRIOR - A new member record that becomes associated with a given owner record is inserted into the logical sequence of member records immediately **before** the current member record of the set.

Example: Suppose the member order of HAS is PRIOR. Suppose department 101 has four employee record occurrences and the third one is presently the current member of HAS. The next employee record occurrence to become associated with department 101 will enter the logical sequence of employee records immediately before the third one.

IMMATERIAL - This order option should be used if the application designer does not care about the set order. This set ordering minimizes the time needed to insert records into a set.

SORTED - The member record occurrences for a given owner record occurrence are logically sorted on the basis of one or more data items in the member record type. These data items are collectively called the member **sort key**. The sorting for each data item in a sort key may be on an ascending or descending basis. In the DDL, the application designer can also specify how MDBS.DMS should handle member records with duplicate sort key values. The options are to prohibit duplicates or to incorporate them on a FIFO, LIFO or IMMATERIAL basis. Finally, the designer can optionally specify how many bytes of a sort key are to be used in maintaining the set's member order index.

Example: Suppose the member order for the HAS set is sorted on last name, first name, and ID (all on an ascending basis). Then all employees in a given department can be accessed according to this sorted order. It also permits MDBS.DMS to very quickly find the EMPLOYEE record occurrence that has a particular sort key value.

These six ordering options have been described for a set's member order. The same options are available for a set's owner order. For instance, the NEEDS set in Figure II-2 might have as its owner order: sorted on JOBCODE. The member order could be sorted on SKILCODE. The result is that all jobs for a particular skill can be accessed in sorted order on the basis of JOBCODE values. All skills for a given job can be accessed in sorted order on the basis of SKILCODE values.

To have MDBS.DMS maintain two (or more) different orderings, two (or more) sets between the two record types can be defined. For example, if you want to have the employees in a department maintained in sorted order on the basis of the ID field and to be maintained in sorted order on the basis of LASTNAME, then two l:N sets would be defined in place of HAS. The first member order is sorted on the ID field. The other member order is sorted on LASTNAME.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

Special Ways of Using Sets

To this point a set has been used to represent a relationship between two different record types defined by the application designer. There are four other ways for using sets: SYSTEM-owned sets, recursive sets, sets with multiple member record types, and sets with multiple owner record types. In the last three situations, all four kinds of sets are allowed. In the case of SYSTEM-owned sets, only 1:1 and 1:N sets are permitted. There is also a special 1:N set called $SYSSET that is automatically defined by MDBS.DDL when the data base is initialized.

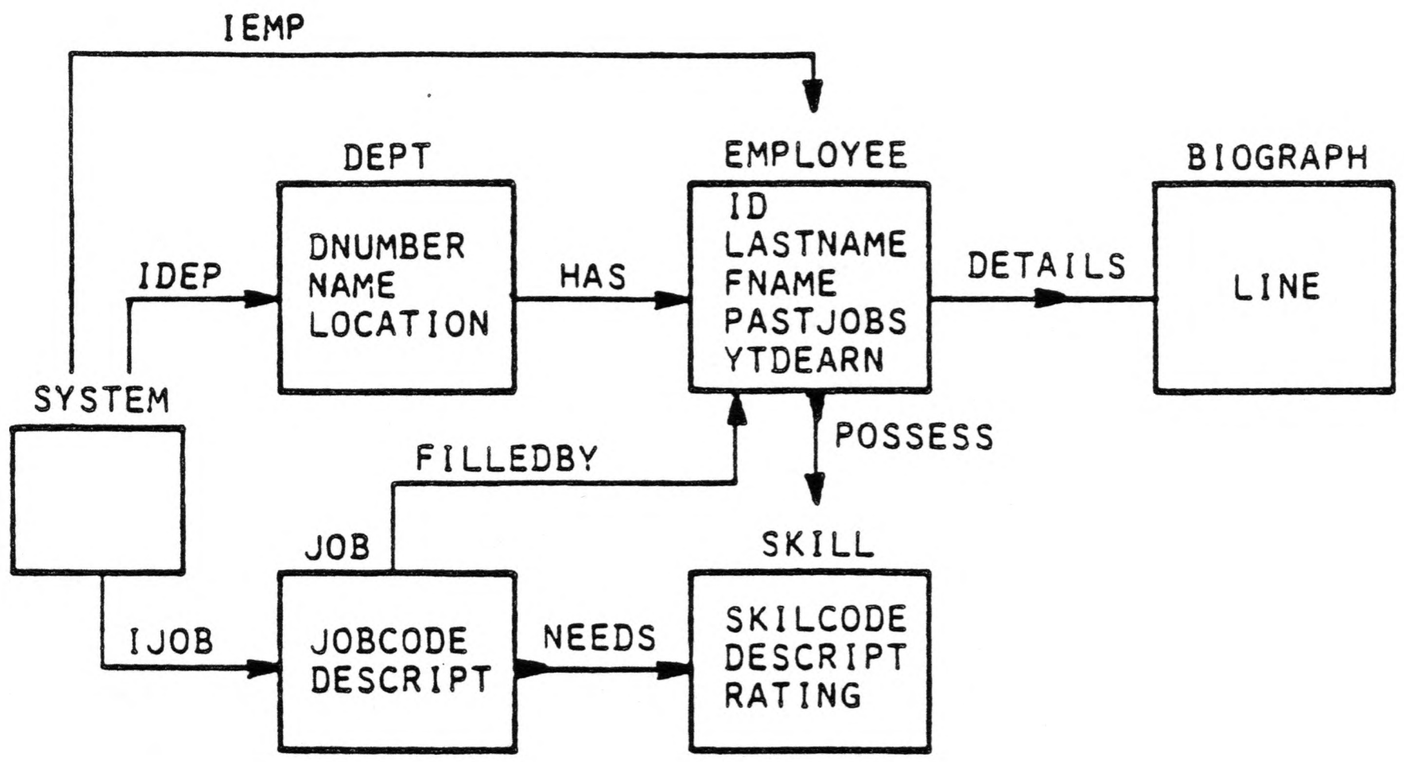


Figure II-3. System-owned Sets

System-owned set. When a data base is initialized, MDBS.DDL automatically creates a special record type called SYSTEM. This special record type has one record occurrence in the data base; the SYSTEM record occurrence is also created when the data base is initialized. It contains no data. A system-owned set is a set that has SYSTEM as its owner and some other record type as its member. Figure II-3 shows three system-owned sets: IEMP, IDEP, IJOB. A record type can be the member of zero, one, or more system-owned sets.

Declaring a sorted system-owned set causes MDBS to automatically form and maintain an index into the occurrences of the member record type. For efficiency, this is actually a multi-level, balanced index. The index is maintained in accordance with the member order declared for the system-owned set (e.g., FIFO, LIFO, SORTED, etc.). Suppose that IEMP is sorted on LASTNAME and FNAME. Then a sorted list (on last and first names) of employees can be quickly produced through IEMP. This also allows MDBS.DMS to quickly find a particular employee occurrence on the basis of its sort key value (i.e., a last and first name). Many system-owned sets can be declared for the same member record type. A system-owned set incurs very little overhead relative to most other kinds of sets.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

Recursive set. A set is recursive when it has the same record type as both owner and member. It is a means for relating occurrences of a record type to other occurrences of the same record type. For example, an employee can manage many other employees who, in turn, manage still other employees, and so forth. Thus there is a one-to- many relationship among employees. This recursive situation is easily handled by the recursive set, MANAGES, that is shown in Figure II-4. Just like any other l:N set, MANAGES will have a member order. The need for recursive l:N and N:M sets is encountered in many application areas (e.g., the chart of accounts in accounting applications). Recursive sets are not allowed in flat file, hierarchical, shallow network, or CODASYL network systems.

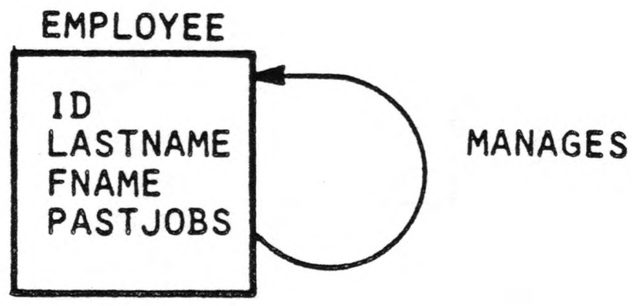


Figure II-4. A Recursive Set

Multiple Member Forked Sets. Just as with CODASYL networks, MDBS extended networks permit l:N sets that have multiple member record types. With MDBS, multiple member record types are also allowed for l:l, N:l, and N:M sets. Figure II-5 gives an example of a forked 1:N set with multiple member record types. This set is called CONTAINS.

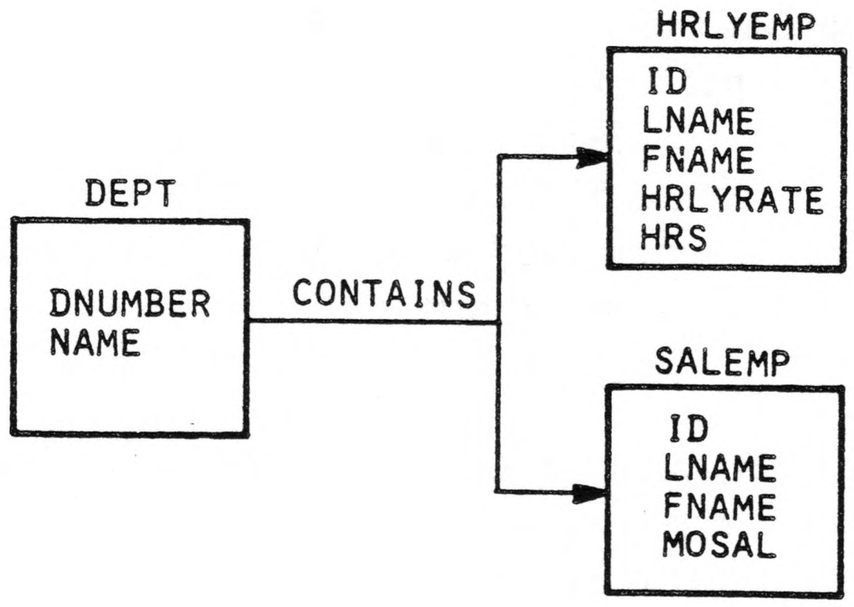


Figure II-5. Multiple Member Forked Set

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

Because this is a l:N set, a record occurrence of DEPT can have many member record occurrences, some of the type HRLYEMP and some of the type SALEMP. Note that the two member record types have some data items in common and others that are different.

All of the usual set orders are allowed with a multiple member set. For instance, CONTAIN could be sorted (ascending) on LNAME and FNAME. The employees in a given department can then be accessed alphabetically, regardless of whether they are salaried or hourly workers. In addition, the record type name could be included as part of the sort key. If CONTAINS is sorted first on (ascending) record type name and then on LNAME and FNAME, then for a given department, an alphabetical list of hourly employees followed by an alphabetical list of salaried employees could be accessed. A maximum of 127 member record types are allowed for any set.

Multiple Owner Forked Sets. With MDBS, more than one record type is allowed as the owner of any set. An example of a forked N:M set with multiple owner record types is shown in Figure II-6. It indicates that many skills can be possessed by an hourly or salaried employee. Conversely, a skill may be possessed by many employees. If the member order for POSSESS is sorted (ascending) on SKILCODE, then the skills for each employee (salaried or hourly) can be accessed in sorted order. If the owner order for POSSESS is sorted (ascending) on ID, then all employees having a given skill (regardless of whether they are salaried or hourly) can be accessed in order according to ID. A maximum of 127 owner record types are allowed for any set.

$SYSSET Set. This is a special FIFO l:N set that MDBS.DDL automatically defines whenever a data base is initialized. Its owner record type is SYSTEM. Its member record type(s) is (are) automatically and dynamically established (and re-established) by MDBS.DMS during the execution of an application program. The member record occurrences of $SYSSET can be records inserted manually by the application program or inserted by a Boolean DML command..

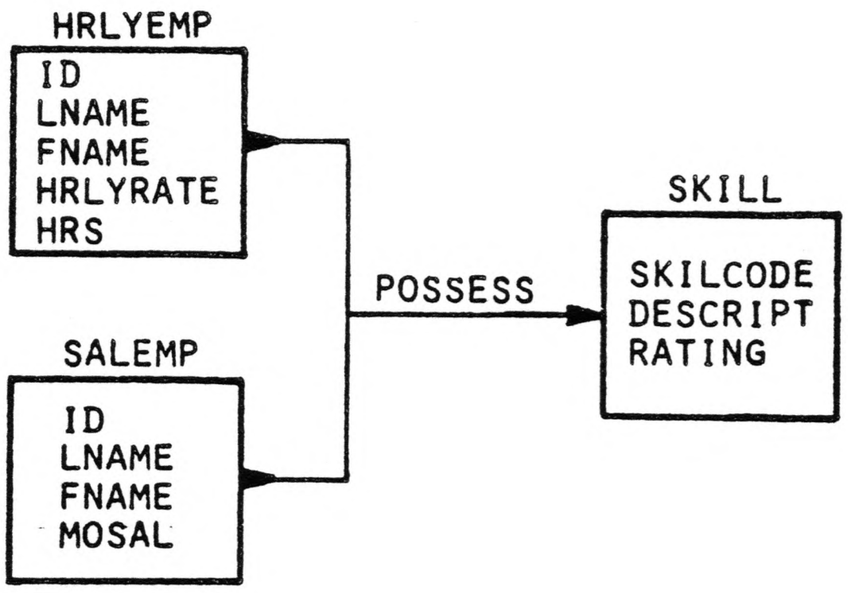


Figure II-6. Multiple Owner Forked Set

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

Set Retention

Any set in the DDL source can be declared to have a FIXED or OPTIONAL set retention. If the set retention is FIXED, then any connection that is made between an owner record occurrence and a member record occurrence is permanent. Although either record could be deleted, they cannot be disconnected from each other without deletion. OPTIONAL set retention means that there is no restriction on disconnecting an owner record from a member record; the two records can be disconnected without deleting either record.

Set Insertion (Connection)

Inserting a member (or owner) record into a set means that it is being connected to an owner (or member) record. The member record type(s) of a set can be declared to have either automatic (AUTO) or MANUAL set insertion. AUTO means that whenever an occurrence of the member record type is created in the data base, it is automatically connected to an occurrence of the set's owner record type(s). MANUAL means that whenever a member record is created in the data base, a DML command can be used to connect it to an occurrence of the set's owner record type(s). The owner record type(s) of a set can also be declared to be either AUTO or MANUAL. AUTO means that whenever an occurrence of the owner record type is created in the data base, it is automatically connected to an occurrence of the set's member record type(s). MANUAL means that this connecting is not automatic, but can be accomplished as needed with a DML command.

TYPES AND SIZES OF DATA ITEMS

Each data item declared with the DDL must be given a type and size. MDBS supports nine different types of data items. Data item size refers to the length of an occurrence of the data item. Permissible data item size depends on the type of the data item. The nine types of data items are:

binary	integer	string
character	internal decimal (idec)	time
date	real	unsigned

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

No size is explicitly stated in the DDL source for date or time data items. The size for an internal decimal data item is stated in terms of the total number of digits to be used (the number of digits to the right of the decimal point is also stated). Sizes for the other six types of data items are stated in terms of bytes. Automatic data compression occurs for string, binary, date, and time data items. Suppose FNAME is a string data item of 12 bytes. When the item occurrence 'Bob' is stored, MDBS allocates fewer than 12 (not 12) bytes. Suppose LINE in Figure II-2 is a string data item of size 250 that repeats five times. Thus a maximum of 1250 bytes are allowed in a LINE occurrence. If only the first 375 bytes are nonblank in a given LINE occurrence, MDBS allocates only about 375 bytes for that occurrence.

Table II-1 shows the minimum and maximum size (n) that can be specified in the DDL for each type of data item. The corresponding storage requirement (in bytes) for a specified n is shown in the table. For example, an occurrence of an IDEC data item of size n=8 will occupy 5 bytes of storage. The maximum range of values for each type of data item is also given. These are maximums with respect to the data base. A host language may impose additional constraints (see MDBS System Specific Manuals). For instance, a data item that is integer of size n=2 can take on any value in the range:

$$-32768 \text{ to } 32767 \ (-2^{8n-1} \text{ to } 2^{8n-1}-1 \text{ where } n=2).$$

Further data item range examples are shown in Appendix E.

The following are guidelines for determining what type to declare for a data item. If the data item's values can take on positive and negative integer values, then an appropriate type declaration is integer. If the values are integers, but never negative, then unsigned is a proper choice. The selection of real versus internal decimal representation for a data item is largely dependent on the host language(s) being used (see the appropriate system specific manuals). For a host language that supports both real and internal decimal variables, real is used for data items with floating point values; if great precision is required for a data item's values, in order to avoid small round-off errors during host language computations, then the internal decimal (IDEC) type should be used. The binary type is used when a data item's values need to be manipulated in a binary form within a host language program (e.g., representation of graphical images, voice images, etc.). If a data item's values are times or dates, then the time or date type (respectively) should be used. This takes advantage of MDBS data compression. The exceptions are where the times cover more than 256 hours, where dates range across more than 126 years, or where dates prior to 01/01/0000 are needed. In such cases, the character data type can be used.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

Type	Min Size(n)	Max Size(n)	Storage requirement in bytes (n=specified size)	Maximum Range (n=specified size)
integer n	1	16	n	-2^{8n-1} to $2^{8n-1}-1$
real n	2	16	n	$-2^{127}(1-256^{1-n})$ to $2^{127}(1-256^{1-n})$
binary n	1	65535	typically < n	00000000 to 11111111 (binary) for each byte
unsigned n	1	16	n	0 to $256^{n}-1$
idec n(digits)	1	30	((n+1)/2)+1 rounded down	$-10^{63}(1-10^{-n})$ to $10^{63}(1-10^{-n})$ if n is even $-10^{63}(1-10^{-n-1})$ to $10^{63}(1-10^{-n-1})$ if n is odd
character n	1	65535	n	any legitimate character (usually 0 to 255) for each byte
string n	1	250	typically < n	any character (except control characters) for each byte
time	--	--	3	00:00:00 to 255:59:59
date	--	--	2	any 126 year period

Table II-1. Types of Data Items

For character data there is a choice of using either the character type or the string type. If the string type is selected, data values are compressed while in the data base, through the elimination of trailing blanks and the compression of consecutive repeating characters. Of course, these values are restored to their original form when extracted from the data base. For the string type, any control characters (including tabs) are permanently transformed into blanks. For alphanumeric data values, (especially for text data) the string type is recommended if there are numerous instances of consecutive repeating characters. If there are very few consecutive repeating characters or if n is small (less than 7), then the character type is recommended. If a string type is declared, but no compression is possible for a data value, then the storage space used for that data value is n + 2 + n/127 (rounded down). This is also the worst case storage space requirement for a binary data type (occurring when a binary value cannot be compressed). The best case storage requirement for a value of a string or binary data item is 1 byte.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

With the exception of binary, any of these types of data items may appear in a sort key. Except for binary data items, any data item can have a feasibility range specified with the DDL. When a feasibility range for a data item exists, the data base control system will automatically perform appropriate data integrity checks whenever attempts are made to store or modify an occurrence of the data item.

When a data value is stored into a data base, it is automatically converted into an internal MDBS representation. This internal representation is the same, regardless of the host language of the application program that loaded the data. When data is retrieved, it is automatically converted from its internal MDBS representation into the representation used by the host language of the retrieval program (or by the query system). Thus application programs in many different host languages can access the same data base. The conversion between internal MDBS representation and a host language representation is described in the system specific manual for that host language. Since most host languages do not support all of the MDBS types, two different MDBS types may be mapped into a single variable type in the host language during data storage and retrieval. These mappings are explained in the system specific manuals. An application developer who intends to use multiple host languages should be aware of the different language mappings and should select a data item's type and size on the basis of the most restrictive host language mapping.

Note

If you use the default language specification (English), a character data item that is used as a sort key must be in the ASCII range of 0 to 127 to maintain correct sort order. If a character with an ASCII value greater than 127 is used, the record order within the set index will be incorrect.

CONTROLLING RECORD PLACEMENT

As mentioned earlier, a record type (and therefore its occurrences) can be assigned to one or more areas. This gives the application developer some control over the physical placement of records. An even greater degree of control is allowed. With MDBS, the application developer has three options for **governing** the **placement** of a record occurrence **within its allotted area(s)**: CALC, clustering, and system default.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

CALC. Any data item or group of data items within a record type can be declared to be the **calc key** for that record type. The calc key value for an occurrence of that record type is input to a hash routine that controls the physical placement of the record occurrence in the allotted area(s). The application designer has the option of prohibiting duplicate calc key values, if desired. If a record type has a calc key, any of its record occurrences can be directly accessed on the basis of a calc key value. Note that a record type can have a calc key and can also be the member of one or more system-owned sets.

MDBS has several calc algorithms. If the first algorithm yields a page with insufficient space for the record occurrence being created, the second algorithm is used, and so forth until a page with sufficient space is found. If none of the algorithms calculates a page with sufficient space, then a spillover approach to overflow is used commencing with the first page following the page computed by the last algorithm. The spillover is restricted to the area in which the last computed page is located and is a wrap-around spillover within that area. If no pages in the area have sufficient space, then MDBS.DMS issues an error indicating that there is insufficient room.

Clustering. In the DDL source specification, an application designer can request MDBS to cluster member (or owner) records close to the owner (or member) record(s) with which they are associated (i.e., connected) via an indicated set. The purpose is to have all member (or owner) records for a particular owner (or member) record clustered in such a way that they can all be resident in central memory at once. This is useful (from the standpoint of minimizing disk I/O) if it is often necessary to access all member (or owner) records whenever an owner (or member) record is accessed. Note that clustering is comparable to the VIA SET record location mode proposed for CODASYL data base management systems, except that it is also available for N:M sets in MDBS.

System default. If neither CALC nor clustering is selected to control record placement within an area(s), then MDBS will control this placement based on any area limitations (within the clause).

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Definitions and Examples

DATA SECURITY

In addition to data encryption and user passwords, highly selective data security is provided by an access code technique. This is fully described in Chapter IV.

MISCELLANEOUS

Comments are allowed in the DDL source for documentation purposes. Each data base, area, record type, set, and data item name can be given synonyms and/or a title. A synonym can be substituted for the original name when issuing a QRS query. Titles and synonyms are incorporated into the data dictionary. Titles can be used to provide comments about a data item, set, etc., for users of the data dictionary. The data dictionary can be interactively queried via the QRS and IDML modules.

© Copyright 1987 Micro Data Base Systems, Inc.

III. MDBS.DDL SPECIFICATIONS

NOTATIONAL CONVENTIONS

After the logical structure for a data base has been drawn out, as in Figure II-2, it can be formally specified with the MDBS Data Description Language. A sample data base description on an introductory level is shown in Figure III-1. An elaboration of that schema description, incorporating many advanced features, is shown in Figure III-2. This chapter gives a detailed explanation of the DDL.

A DDL source specification consists of several sections. Each section consists of various kinds of clauses; some of which are optional and some of which are required. These clauses are presented on a section-by-section basis. Discussion of those clauses that can be regarded as advanced features are denoted in the left margin by a vertical bar. The complete BNF syntactic description of all clauses appears in Appendix B. Figure III-3 shows the meanings of the notations used.

A clause is composed of one or more of the following **terms**: keywords, identifiers, file names, strings, integers, user names, and passwords.

Keyword A keyword is denoted by upper case letters; all other terms in a clause are in lower case. This distinction is for documentation purposes only. Upper and lower case characters can be mixed in an actual DDL source specification. When the data base is initialized, all lower case characters are mapped into upper case characters in the data dictionary, with the exception of characters enclosed in double quotes (""). A complete list of DDL keywords appears in Appendix A.

Identifier An identifier is the name of a data base element (a data item, record type, set, or data base area) selected by the application developer. An identifier consists of one to eight characters with the first character alphabetic. The remaining characters may be alphabetic, numeric, or an underscore (_). In this manual, DDL sections denote identifiers by the following abbreviations:

id-n is a new, unused identifier (such as **id-1** and **id-2**).

di-n identifies a previously defined **d**ata **i**tem (such as **di-1** and **di-2**)

rt-n identifies a previously defined **r**ecord **t**ype (such as **rt-1** and **rt-2**)

st-n identifies a previously defined **s**et name (such as **st-1** and **st-2**)

ar-n identifies a previously defined **ar**ea name (such as **ar-1** and **ar-2**)

A DDL keyword can also be used as an identifier, by using one of the following DDL keywords prior to the identifier: IS, ARE, WITH, OF, TO, BY. A list of MDBS-QRS keywords is in Appendix G. These QRS keywords should not be used as identifiers.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

```
/******** sample data base description (introductory level) ************/
/********************** data base identification **********************/
database name is JOBS
      file "JOBS.DB", size is 300 pages, page size is 512 bytes
/******************* user and password definitions *******************/
user is "BOB SMITH" with GTC
user is ANALYST      with7778XK4
/************************ record type defintions ***********************/
record name is DEPT
      item name is DNUMBER       integer 1
      item name is NAME          character 12
      item name is LOCATION      string 35
record name is EMPLOYEE
      item ID                    character 9
      item LASTNAME                 string 20
      item FNAME                   string 12
      item PASTJOBS                 string 25 occurs 3 times
      item YTDEARN                  idec 7,2
record BIOGRAPH
      item LINE                  string 50 occurs 5 times
record SKILL
      item SKILCODE                 integer 2
      item DESCRIPT                 string 55
      item RATING                  real 2
record JOB
      item JOBCODE                  integer 2
      item DESCRIPT                string 30
/************************** set definitions ****************************/
set name is IEMP
            type is l:n
            owner is SYSTEM
            member is EMPLOYEE order is sorted
                                    by ascending (LASTNAME, FNAME)
set name POSSESS     type is n:m
            owner is EMPLOYEE order is sorted by az ID
            member is SKILL     order is immaterial
set DETAILS            type is l:l
            owner is EMPLOYEE, member is BIOGRAPH
set FILLEDBY           type is l:n
            owner is JOB, member is EMPLOYEE order is fifo
set NEEDS              n:m
            owner JOB sorted descending JOBCODE
            member SKILL sorted ascending (RATING,SKILCODE)
set HAS                l:n
            owner DEPT, member EMPLOYEE order lifo
set IDEP               l:n
            owner SYSTEM       member DEPT
set IJOB               l:n
            owner SYSTEM       member JOB fifo
end
```

Figure III-1. Sample DDL Source Text

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

```
/*****
 *
 * sample data base description (advanced level)
 *
 *****/

db JOBS          file "JOBS.DB"
                 size 300 pages, page size 1024
                 logfile "JOBS.LOG"

/********************** define defaults for item types ******************/
    default for unsigned 2
    default for str 50

user "BOB SMITH" with GTC       read access (a,b) write access a
user ANALYST with 7778XK4       read (b-e) write (b-f)
user "K FERGUSON" with "tashi" access (b,p)
user "D LEHR" with "smiles" access (a-p)

/** define additional areas to supplement the main data base area ***/

area name is job1
      file name is "JOB1.DB"
      size is 100 pages, pointers not allowed
      page size is 512 /* Note:  this page size would not be allowed in Version 3a
                                 because the main area's page size is 1024 */
      read access is (a-d) write access is (a,p)
area JOB2         file "JOB2.DB"
                  size 700 pages
      read (b,d)  write (a,b,f,p)

/********************** record type definitions **********************/

record DEPT
      in JOBS calc key is NAME nodup
      read access b write access (a,b)
            item DNUMBER int 1
                        range is 1 to 42
                        syn is DNO
            item is NAME char 12    encrypted
                        write access b
            item LOCATION str 35    syn LOC
                              range "A" to "ZZZZZZ"
record employee in JOB1, key is ID nodup
      read access (a,d) write access (a,p)
            item ID char 9 encrypted range is "0" to "999999999"
            item LASTNAME str 20        range "Aa" "Zz"
            item FNAME     str 12       range "Aa" "Zz"
            item PASTJOBS str 25 occurs 3 times
            item YTDEARN idec 7,2 encrypted range 0.00 94000.00
                  read access a write access a
```

Figure III-2. Sample DDL with Advanced DDL Features

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

```
record BIOGRAPH in JOB2
      read access b write access p
            item LINE str occurs 5 times
record SKILL in (JOBS,JOB2) calc key is SKILCODE nodup
      read access (b,d) write access f
            item SKILCODE unsigned syn SC range 0 to 3000
            item DESCRIPT str 55
            item RATING real 2                    range 0.0 to 4.0
record JOB in any area
      read access b write access (a,p)
            item JOBCODE unsigned 1               range 1 to 250
            item DESCRIPT str
/************************** set definitions ***************************/
set IEMP, type l:n, retention fixed
      read access d write access p
            owner SYSTEM
            member EMPLOYEE, order sorted by ascending (lastname, Fname)
                  insertion auto
set POSSESS, type n:m
      read access (a,d) write access (f,p)
            owner EMPLOYEE, order sorted by az ID
            member SKILL
set DETAILS, type l:l, fixed
      read access (b,d) write access p
            owner EMPLOYEE
            member BIOGRAPH auto
set FILLEDBY, type l:n
      read access (a-d) write access p
            owner JOB
            member EMPLOYEE fifo auto
set NEEDS n:m
      read access b write access (f,p)
            owner JOB sorted za JOBCODE
            member SKILL sorted az (RATING,SKILCODE)
set HAS l:n
      read access (a,b) write access a
            owner DEPT
            member EMPLOYEE lifo auto
set IDEP l:n, read access b
      owner SYSTEM    member DEPT auto
set IJOB l:n, read access b
      owner SYSTEM    member JOB auto fifo
END
```

Figure III-2. Sample DDL with Advanced Features (continued)

(See Appendix H for an alternative layout of this DDL specification.)

File names A file name in a DDL section is denoted by the word **file** followed by a numeral (such as **file-1** and **file-2**). File names are chosen by the application developer; they must be fully qualified file names within the host operating system.

Integers An integer is a whole number that specifies the size of a data base element; integers are denoted by the abbreviation **int** followed by a numeral (such as **int-1** and **int-2**).

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Strings A string is a contiguous group of characters enclosed within double quotes. Currently, strings are used only for the title clause in a data base specification. Strings are denoted by the word string followed by a numeral (such as **string-1** and **string-2**). Each string within a DDL section has a unique number as its suffix. The actual string values are chosen by the application designer. The declaration of a string value (with quotes) must fit on one line of the DDL source specification.

User name A user name, denoted by **usr**, is one to sixteen characters (alphanumeric).

Passwords A password, denoted by **pass**, is one to twelve characters (alphanumeric).

————	An underlined expression must appear in the clause.
[ ]	Zero or one of the alternatives within the brackets can appear in the clause.
{ }	One of the alternatives within the braces must appear in the clause.
{ }*	One or more of the alternatives within the braces can appear in the clause.

Figure III-3. Notation

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

DDL SECTIONS

A DDL source specification consists of up to nine kinds of sections. These sections must appear in the order given below. Clauses within a section are stated in a free format (i.e., column positioning is unimportant for the DDL analyzer). Upper case and lower case can be used interchangeably. More than one clause can appear on a line; a clause (but not a term) can be split over more than one line; unless otherwise indicated, optional clauses within a section can appear in any order. Clauses can be separated by a comma or blanks. A section is terminated with a blank or period. Comments can appear between any two terms or clauses within a DDL source specification.

The kinds of DDL sections and their ordering are as follows:

Identification Section
User Section
Area Section (optional)
Record Sections
    Data Item Sections within each record section (optional)
Set Sections (optional)
    Owner Section for each set section
    Member Section for each set section
End Section

The identification section identifies the data base. The user section indicates the possible data base users (or user classes) and gives a password to each. The optional area section is used to define extra areas if they are desired. There follows a series of record sections. There is one record section for each record type in the logical structure. Associated with each record section is a data item section for each of that record type's data items. If a record type has no data items, then the corresponding record section has no data item sections. After all record types (and associated data item types) have been defined in the record (and data item) sections, all sets in the logical structure are defined in the set sections. There is one set section per set. Associated with each set section is an owner section indicating the set's owner(s) and a member section indicating the set's member(s). A DDL source specification always terminates with the end section.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Comments

Comments can be located anywhere in a DDL source specification. All comments begin with the two character sequence /* and end with the two character sequence */. All text between these character sequences are considered comments and bypasses the analysis of DDL. Comments can extend over many source text lines.

Examples:

```
/* THIS IS A SINGLE LINE COMMENT */

/*****
 *
 * DDL SOURCE SPECIFICATION  (MULTIPLE LINE COMMENT)
 *
 *****/

/*
      RECORD TYPE DEFINITIONS  (MULTIPLE LINE COMMENT)
*/
```

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Identification Section

The identification section must be present. This section is used to give a name to the data base. In addition, the data base designer can optionally specify:

- the physical file that will hold the data base
- the number of pages in the data base
- the size of a data base page
- the physical file that will hold the logged transactions
- a title for the data base
- synonyms for the data base name
- data item size defaults for the DDL specification
- the language (English, French, German, etc.) to be used as a basis for sorting

Required Clause

{ DATABASE / DB } NAME IS id-1

The data base name is indicated by identifier id-1. A data base area named id-1 is automatically created when the data base is initialized. This is referred to as the **main area** of the data base.

Examples:

```
DATABASE NAME IS MED
DB NAME IS SCHOOL
DATABASE PILOT
DB PILOT
```

Optional Clauses

a) File clause FILE NAME IS "file-1"

The main data base area will physically reside on the disk file indicated by file-1. MDBS.DDL will automatically create file-1 during data base initialization. File-1 is a fully qualified file name for the host operating system.

Default: based on the data base name id-1, but operating system dependent

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Examples:

```
FILE NAME IS "SCHDB"
FILE "INVEN/DB:1"
FILE "B:DEB"
FILE "/usr/db/customer.db"
```

b) Page clause SIZE IS int-1 PAGES

The number of pages in the area id-1 will be int-1. If the environment allows 16 areas, the maximum size is 4095 pages. If only 8 areas are allowed, the maximum size is 8191.

Default: operating system dependent (typically 50)

Examples:

```
SIZE IS 35 PAGES
SIZE 215
SIZE 1000 PAGES
```

c) Page size clause

PAGE SIZE IS int-2 [BYTES | WORDS]

The size of each page in the area id-1 is int-2 bytes or words.

Notes: MDBS version 3a requires int-2 to be a multiple of 256. With MDBS version 3c, int-2 need not be a multiple of 256, but must be at least 256.

MDBS version 3a requires all data base areas to have the same page size (int-2). With MDBS version 3c, the main data base area has the largest page size (int-2); other areas can have different page sizes, but no area can have a page size that exceeds int-2.

Defaults: i) operating system dependent (typically 512 bytes)
ii) Unless WORDS is specified, the page size is in bytes

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Examples:

```
PAGE SIZE IS 256 BYTES
PAGE 1024 WORDS
PAGE 1024
```

d) Log file clause

{ LOG FILE / LOGFILE } NAME IS "file-2"

Transactions will be logged to the file indicated by file-2. MDBS.DDL will automatically create file-2 during data base initialization. This file is reinitialized by the MDBS-RTL recovery utility (called RCV). File-2 is a fully qualified file name in the host operating system.

Default: no log file for transaction logging

Examples:

```
LOGFILE NAME IS "MED.LOG"
LOG FILE NAME IS "PILOTL"
LOG "B:DEBLOG"
```

e) Title clause TITLE IS "string-1"

The data base is given the title indicated by string-1.

Default: no title

Examples:

```
TITLE IS "MEDICAL SCHEDULING DATA BASE"
TITLE "PILOT PROJECT:  VERSION 5.2"
```

f) Synonym clause

{ SYN / SYNONYM } IS id-2

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

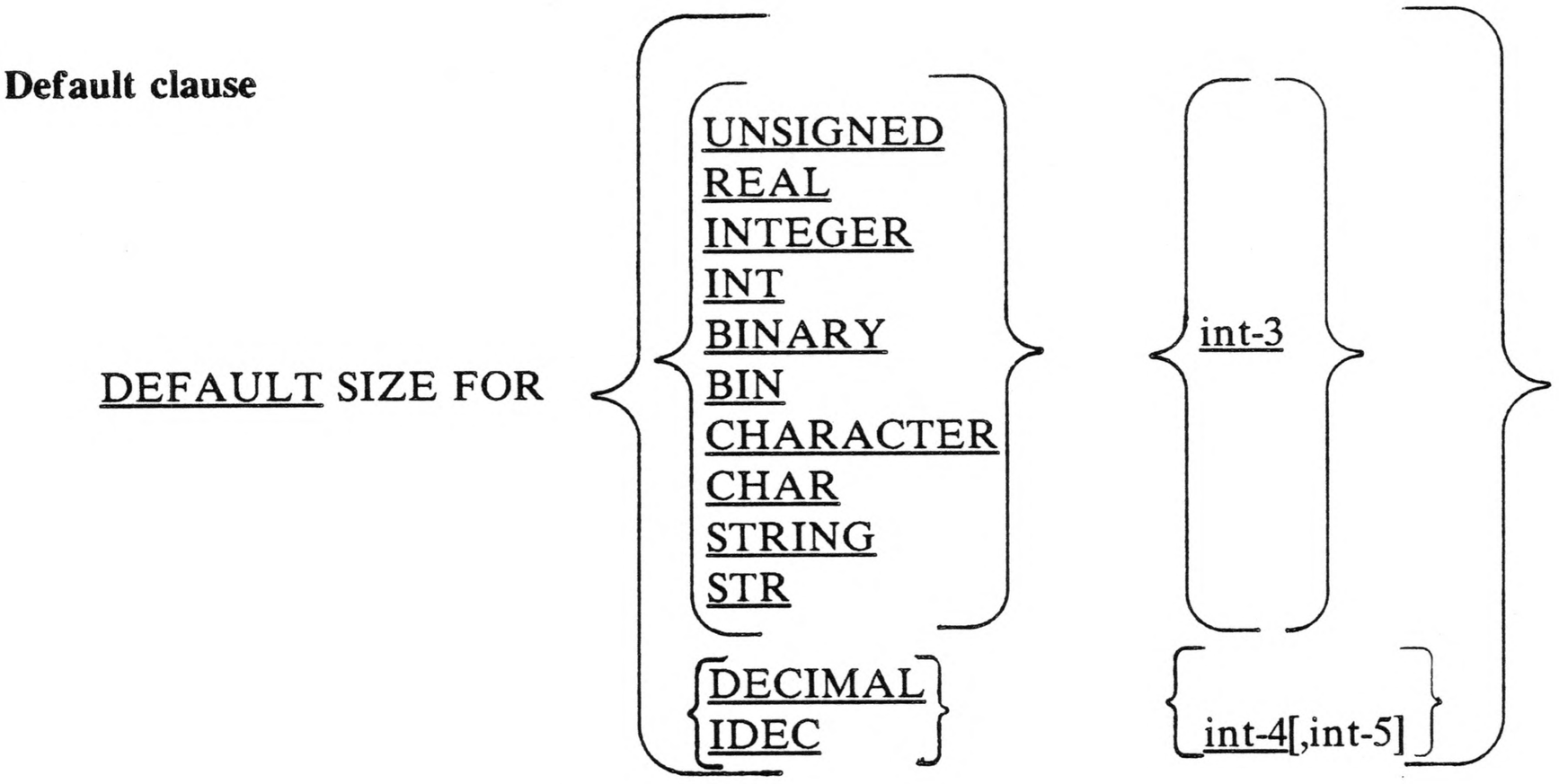
The name indicated by id-2 is a synonym for the data base name id-1. Many synonyms can be defined by repeated use of this clause. MDBS imposes no maximum on the number of synonyms.

Default: none

Examples:

```
SYNONYM IS MEDSCH
SYN S
```

g) Default clause



This clause is used to specify the default size for an indicated type of data item (real, integer, unsigned, etc.). A specified default size for a given type applies to all data items defined within the data base description for id-1. If no size is specified for a data item in a data item section, then the default size stated with this clause is assumed. One default clause can be used for each of the nine types of data items, except for time and date. Default clauses must be the last clauses of an identification section. The default size (in bytes) is indicated by int-3 with the exception of IDEC. For IDEC, int-4 is the total default size (in digits) and the optional int-5 indicates the number of digits to the right of the decimal point. If int-5 is not specified, it is assumed to be zero.

Default: none

Examples:

```
DEFAULT SIZE FOR REAL 4
DEFAULT CHAR 200
DEFAULT IDEC 10,1
DEFAULT DECIMAL 8
```

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

h) Language clause

LANGUAGE IS { DANISH | ENGLISH | FINNISH | FRENCH | GERMAN | NORWEGIAN | SWEDISH }

Any sorting carried out by any MDBS software for the data base's contents will use the normal character collating sequence of the specified language. This includes the maintenance of sorted sets, dynamic sorting of QRS output, and the sorting of report contents produced by programs that RDL generates.

Default: English (that is, standard ASCII sequence)

Examples:

```
LANGUAGE IS FINNISH
LANGUAGE FRENCH
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

User Section

The user section must be present. It is used to give the names and passwords of all data base users. The following clause is used to define each user. For a data base to have five different users (or user groups), five successive clauses of this kind must be specified. No two users can have the same user name. The value of a password for providing data security is discussed in Chapter IV.

Required Clause

USER NAME IS { usr WITH pass / "usr", "pass" }

The user whose name is *usr* has the password *pass*. If a user name or password is not enclosed in double quotes, then its alphabetic characters are converted to upper case. When double quotes are used, no case conversion occurs. User names and passwords that contain embedded blanks must be enclosed in double quotes.

Examples:

```
USER IS "Alvin Wade" WITH SECRET
USER GC15 , "DJL"
USER PRESIDENT MONROE
```

Optional Access Clause

An optional access clause can accompany any instance of a USER clause. This access clause is discussed in Chapter IV.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Area Section (optional)

The area section is optional. It is used to define additional areas for the data base. Depending on the environment, a maximum of either 7 or 15 extra areas can be defined in the area section. See the pertinent System Specific Manual.

Required Clause

AREA NAME IS id-1

An area named id-1 is automatically created when the data base is initialized. This kind of clause (together with its optional clauses shown below) is required for each additional area. Two areas should not have the same name.

Examples:

```
AREA NAME IS REALM4
AREA IS PAYROLL
AREA FASTAC1
```

Optional Clauses

a) File clause FILE NAME IS "file-1"

The area named id-1 will physically reside on the disk file indicated by file-1. MDBS.DDL will automatically create file-1 during data base initialization. Two areas cannot be assigned to the same file. File-1 is a fully qualified file name for the host operating system.

Default: File-1 default is based on the id-1 area name, but is operating system dependent, (e.g., if id-1 is TREE, then the default for file-1 could be TREE.DBA).

Examples:

```
FILE NAME IS "B:RAE"
FILE IS "USV/DB/SALES.DBA"
FILE "SCHREC/DBA:2"
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

b) Page clause SIZE IS int-1 PAGES

The number of pages in the area id-1 is indicated by int-1. For environments allowing 16 areas there is a maximum of 4095 pages per area. The maximum is 8191 for environments allowing no more than 8 areas.

Default: operating system dependent (typically 50)

Examples:

```
SIZE IS 89 PAGES
SIZE 200 PAGES
SIZE 77
```

c) Page size clause

PAGE SIZE IS int-2 [BYTES | WORDS]

The size of each page in area id-1 is int-2 bytes or words.

Note: MDBS version 3a requires all data base areas to have the same page size (int-2). With MDBS version 3c, the main data base area has the largest page size (int-2); other areas can have different page sizes, but no area can have a page size that exceeds int-2.

Defaults: i) defaults to the size of the main area
ii) if WORDS is not specified, the page size is in bytes

Examples:

```
PAGE SIZE IS 512 BYTES
PAGE 512 WORDS
PAGE 256
```

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

d) Pointer control clause

POINTERS ARE [NOT] ALLOWED

This clause allows the application designer to control whether or not dynamic pointer indices are allowed within the area. Disallowing dynamic pointer indices in an area, with many CALCed records, can yield higher CALC access performance. **If pointer indices are allowed in an area, then that area must be on-line for all DML commands that alter the data base.**

Default: dynamic pointer indices are allowed in the area

Examples:

```
POINTERS NOT ALLOWED
POINTERS ALLOWED
```

e) Title clause TITLE IS "string-1"

The area is given the title indicated by string-1.

Default: no title for area

Examples:

```
TITLE IS "AREA FOR SALARY INFORMATION"
TITLE "BIOGRAPHICAL DATA AREA"
```

f) Synonym clause

{ SYNONYM / SYN } IS id-2

The name indicated by id-2 is a synonym for the area name id-1. More than one (no maximum) synonym can be defined.

Default: no synonym

Examples:

```
SYNONYM IS BIO
SYN WAGEAREA
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Record Section

One or more record sections must appear in a DDL source specification. Each record section is used to define a record type. If the record type has data items, then data item sections for each data item must immediately follow the record section. Record types can be defined in any sequence within the group of record sections. More than 255 record sections (i.e., record types) per data base are not allowed.

Required Clause

RECORD NAME IS id-1

This clause defines a record type having the name id-1. The name SYSTEM is not allowed. The SYSTEM record type is automatically defined during data base initialization.

Examples:

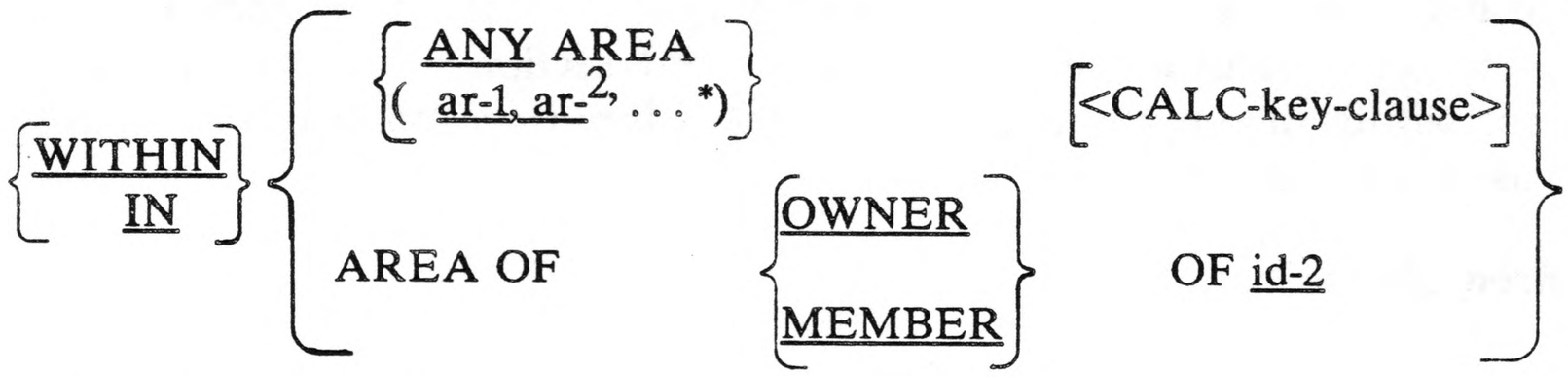
```
RECORD NAME IS SKILL
RECORD IS EMPLOYEE
RECORD DEPT
```

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Optional Clauses

a) Record location clause



This optional clause allows the application designer to control where the occurrences of a record type will be located in the data base. The record type can be assigned to any area or to one or more specified areas (ar-1, ar-2, etc.) with the option of specifying a CALC key for the record type. Alternatively, the record type can be assigned to the area(s) of the owner or member of the set id-2, in which case MDBS will physically cluster those record occurrences that are related to each other through the set id-2. Later in the DDL source specification, id-2 must appear as the name of a set.

Defaults (for record location clause):

i) omission of this clause means that occurrences of the record type can be placed in any defined area

ii) omission of the optional CALC-key-clause means that MDBS determines the placement of record occurrences within the assigned area(s)

Examples:

```
WITHIN ANY AREA
CALC KEY ID NODUP
WITHIN PAYROLL
    KEY (LNAME,FNAME)
IN ANY
IN DBAREA2 DBAREA3
IN (AREA9, AREA7)
   KEY IS JOBCODE NODUP
IN AREA OF MEMBER OF POSSESS
IN AREA OF OWNER FILLEDBY
IN MEMBER POSSESS
WITHIN OWNER FILLEDBY
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

The optional CALC-key-clause, within the record location clause, has the following form:

<CALC-key-clause>:

{CALC KEY / CALC KEYS} [IS / ARE] ({id-3,}*) [NODUP ALLOWED / DUPLICATES ARE [NOT] ALLOWED]

A calc key consists of the data item id-3 or the sequence of data items (id-3, id-4, etc.). The sequence of data items determines the calc key values. Parentheses are not required if there is only one data item in the calc key. All data items that make up a record type's calc key must be defined to be data items for that record type. The application designer can indicate whether or not two record occurrences with the same calc key values are allowed.

Default: duplicates are allowed

Examples:

```
KEY IS ID NODUP ALLOWED
KEY IS (LNAME,FNAME)
KEY IS LNAME FNAME
KEY (ID,LNAME,FNAME) NODUP
```

b) Title clause TITLE IS "string-1"

The record type is given the title indicated by string-1.

Default: no title for the record type

Examples:

```
TITLE IS "EMPLOYEE DATA"
TITLE "SKILL DESCRIPTION AND RATING"
```

c) Synonym clause

{SYNONYM / SYN} IS id-5

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

The name indicated by id-5 is a synonym for the record type name id-1. Multiple (no maximum) synonym clauses are allowed.

Default: no synonym

Examples:

```
SYNONYM EMP
SYN DEP
```

d) Access Clause

An optional access clause can be specified for any record type. This access clause is discussed in Chapter IV.

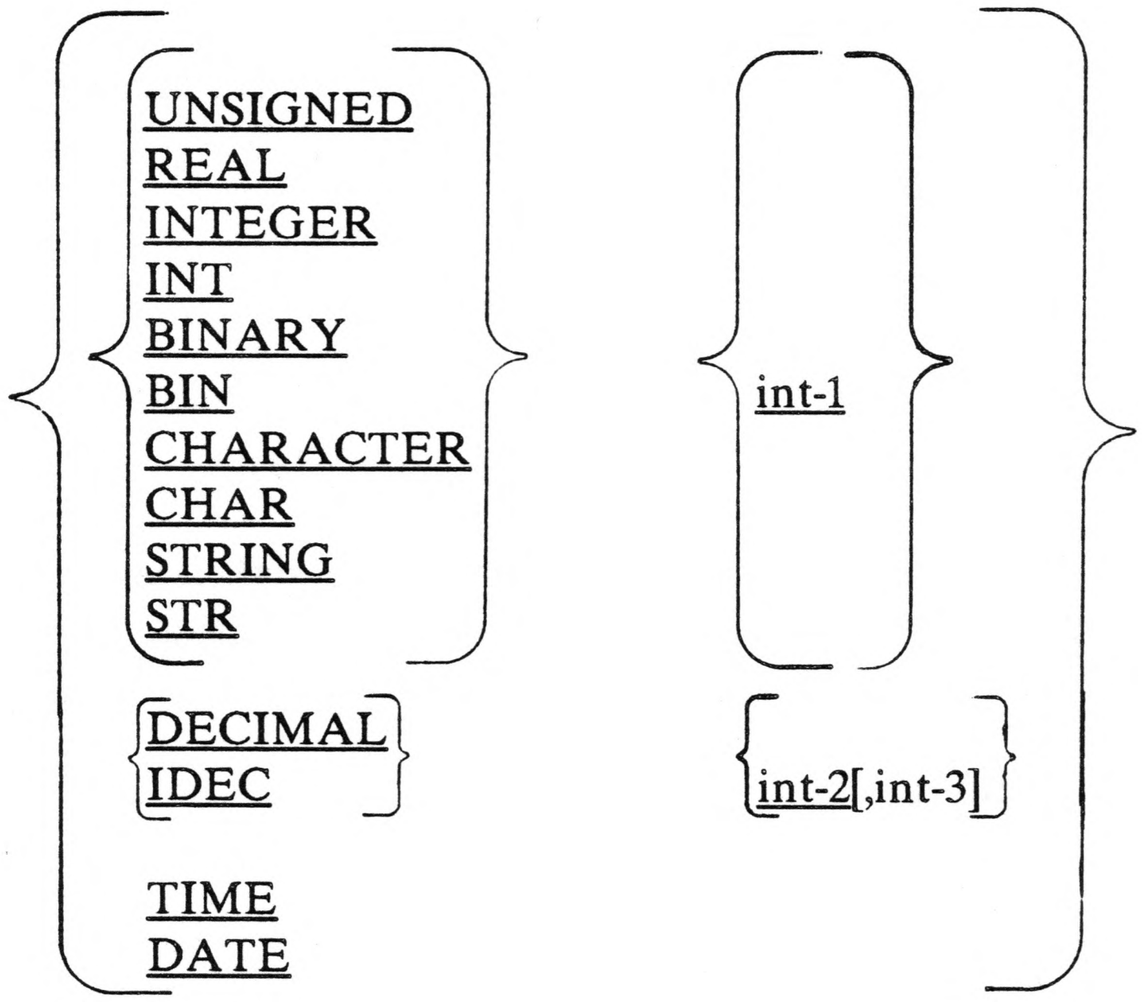
© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Data Item Section

Following each record section is a group of (possibly zero) data item sections. There is one data item section for each data item in a record type. The sequence of data item sections for a given record section is irrelevant to the DDL analyzer. No more than 65535 data items per record type are allowed.

Required Clause



This clause defines a data item having the name id-1. It also indicates the type and size of that data item. Permissible types are UNSIGNED, REAL, INTEGER (or INT), BINARY (or BIN), CHARACTER (or CHAR), STRING (or STR), internal decimal denoted by IDEC or DECIMAL, TIME, and DATE. No size is specified for TIME or DATE. For internal decimal, int-2 indicates the number of digits stored and int-3 indicates the number of digits to the right of the decimal point; int-3 cannot exceed 7. The default for the optional int-3 is 0. For all other types, int-1 indicates the number of bytes for the data item. Maximum and minimum sizes were discussed in the preceding chapter.

DATE data values can be automatically converted to alternative formats by the DBCNV DML command and by the DATFORM QRS command. The DBCNV command can also be used to permit null (i.e., blank) DATE and TIME values.

Examples:

```
ITEM NAME IS NAME CHAR 25
ITEM YTDWAGE IDEC 8,2
ITEM ZIP STRING 9
ITEM BIRTHDT DATE
ITEM NAME ELAPSETM TIME
ITEM AGE INT 2
```

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Optional Clauses

a) Occurs clause OCCURS int-4 TIMES

This clause is used to indicate that the data item is a repeating item. Each occurrence of the repeating item is an array of length int-4. The maximum value for int-4 is 255.

Default: the data item is not a repeating item (in other words, int-4 = 1)

Examples:

```
OCCURS 8 TIMES
OCCURS 35
```

b) Encryption clause IS ENCRYPTED

This clause indicates that values of the data item are stored in the data base in encrypted form. Those values are automatically decrypted when they are retrieved (subject to data security restrictions, discussed in Chapter IV).

Default: no encryption

Examples:

```
IS ENCRYPTED
ENCRYPTED
```

c) Title clause TITLE IS "string-1"

The data item is given the title indicated by string-1.

Default: no title for the data item

Examples:

```
TITLE IS "GROSS PAY"
TITLE "EMPLOYEE ADDRESS"
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

d) Synonym clause { SYNONYM / SYN } IS id-2

The name indicated by id-2 is a synonym for the data item name id-1. Multiple (no maximum) synonyms can be declared.

Default: no synonym

Examples:

```
SYNONYM GP
SYN IS EMPADD
```

e) Access Clause

An optional access clause can be specified for any data item. This access clause is discussed in Chapter IV.

f) Range Clause

The optional range clause, for automatic feasibility checking, is discussed in Chapter V.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Set Section

Zero, one, or more set sections must appear in a DDL source specification. Each set section is used to define a set. A set section must be immediately followed by an owner section, which is immediately followed by a member section. Sets can be defined in any sequence within the group of set sections. MDBS imposes no maximum on the number of set sections in a DDL source specification.

Required Clause

SET NAME IS id-1

This clause defines a set having the name id-1.

Examples:

```
SET NAME IS NAME
SET IS CONTAINS
SET POSSESS
```

Optional Clauses

a) Set type clause

TYPE IS { 1:1 | 1:N | N:1 | M:N | N:M }

This clause indicates whether the set is 1:N, N:M (or equivalently M:N), 1:1, or N:1.

Default: 1:N

Examples:

```
TYPE IS N:M
TYPE 1:N
IS 1:1
TYPE M:N
N:M
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

b) Retention clause RETENTION IS { FIXED / OPTIONAL }

This clause declares the set retention to be either FIXED (a participating record occurrence cannot be disconnected; it can be deleted) or optional (a record occurrence can be connected and disconnected as desired).

Default: optional

Examples:

```
RETENTION IS FIXED
RETENTION FIXED
IS OPTIONAL
FIXED
```

c) Title clause TITLE IS "string-1"

The set is given the title indicated by string-1.

Default: no title for the set

Examples:

```
TITLE IS "MANY-TO-MANY RELATIONSHIP"
TITLE "EMPLOYEE POSSESSES SKILL"
```

d) Synonym clause { SYNONYM / SYN } IS id-2

The name indicated by id-2 is a synonym for the set name id-1. Multiple (no maximum) synonym clauses are allowed.

Default: none

Examples:

```
SYNONYM IS POSS
SYN FBY
```

e) Access Clause

An optional access clause can be specified for any set. This access clause is discussed in Chapter IV.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Owner Section

An owner section must immediately follow each set section. An owner section describes the owner(s) of a set.

Required Clause

{ OWNER IS rt-1
OWNERS ARE (rt-2, rt-3,...*) }

If the set has a single owner record type, its name is indicated by rt-1. If the set has multiple owner record types, those record types are indicated by (rt-2, rt-3, etc.); the ordering of these multiple record type names is irrelevant. No more than 127 record types can own a set.

Examples:

```
OWNER IS SYSTEM
OWNER DEPT
OWNERS ARE (HRLYEMP,SALEMP)
OWNERS (SALEMP,HRLYEMP)
OWNERS REC1 REC2
```

Optional Clauses

a) Set insertion clause INSERTION IS { AUTO / MANUAL }

This clause indicates whether or not the creation of an owner record occurrence causes it to be automatically (AUTO) connected to a member record.

Default: MANUAL

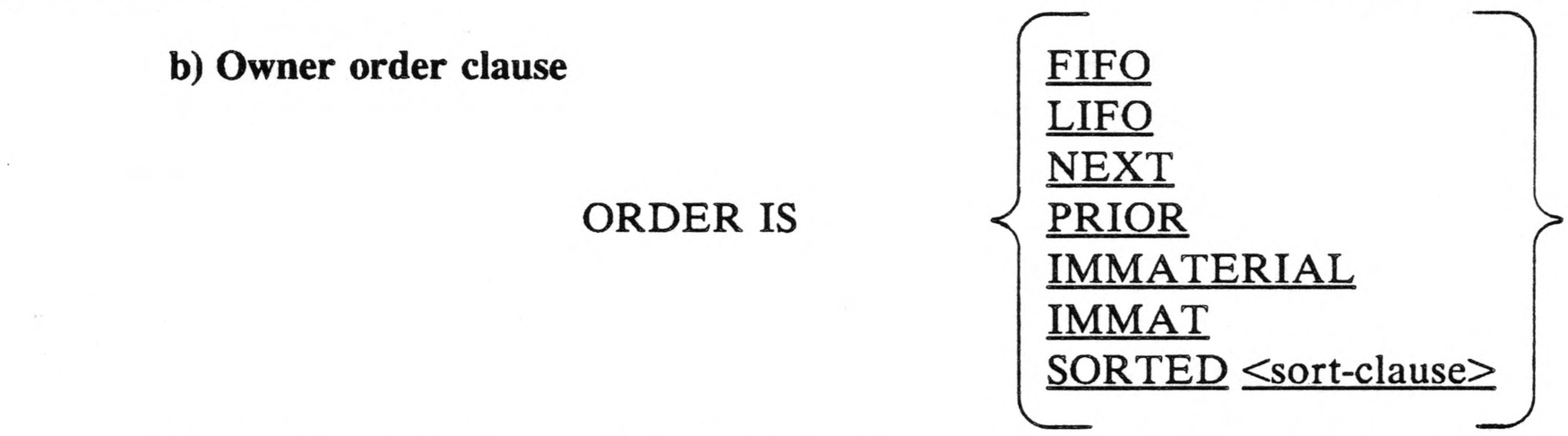
Examples:

```
INSERTION IS AUTO
MANUAL
INSERTION AUTO
```


© Copyright 1987 Micro Data Base Systems, Inc.

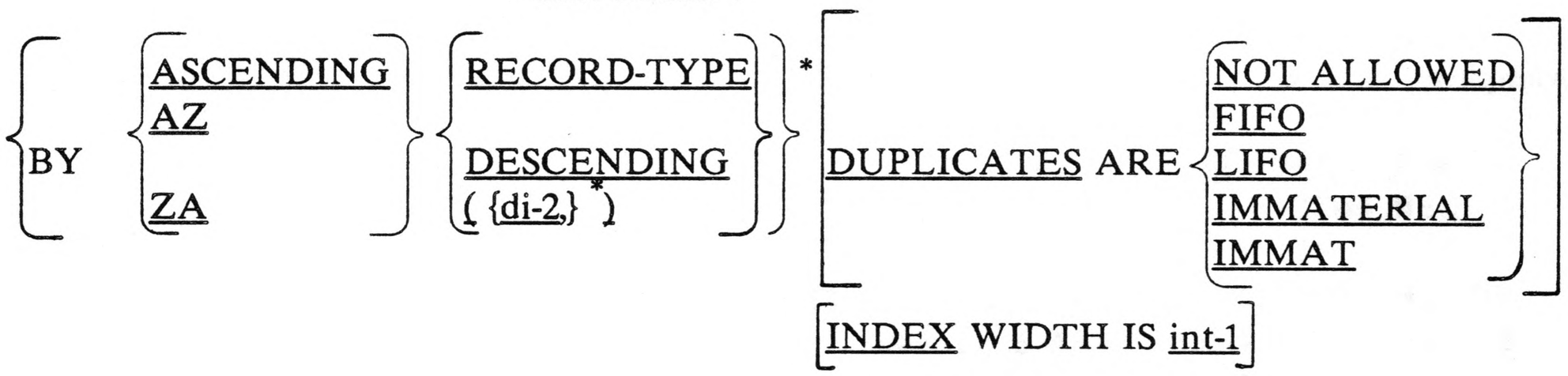
MDBS DDL MANUAL
MDBS.DDL Specifications

b) Owner order clause



This clause indicates the ordering of owner record occurrences that are associated with any given member record occurrence. The permissible orderings are FIFO, LIFO, NEXT, PRIOR, IMMATERIAL (or IMMAT), and SORTED. The sort-clause has the following form:

<sort-clause>:



If the owner order is SORTED, then the sort-clause is required. It indicates the basis for the sorting of owner record occurrences that are connected to a member record occurrence. The sort is made on the basis of one or more data items (di-1, di-2, etc.) that have been defined for the owner record type. A binary data item may not be used as a basis for sorting. If the set has multiple owner record types, then the owner record type names can also be used as a criterion for sorting. Thus a sort key consists of a sequence of data items, plus the possible inclusion of the literal: RECORD-TYPE. The ordering of these components within the sort key definition is from the most important to the least important. The entire sort key can be declared to be ASCENDING (AZ) or DESCENDING (ZA). Alternatively, each component (or group of sort key components) is declared to be either ASCENDING or DESCENDING. The character code of the host operating system determines ascending and descending sequences.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Once the sort key has been specified, the designer can optionally indicate how MDBS should handle record occurrences with duplicate sort key values. The options are: NOT ALLOWED, FIFO, LIFO, IMMATERIAL (or IMMAT). The designer can also optionally indicate how many bytes (int-1) of the sort key are to be used in the multilevel owner index. For instance, if int-1 is 7, then the first seven bytes of the sort key values will be used in maintaining the index. The owner record occurrences are still sorted on the basis of the entire sort key. The index width (of up to 28 bytes) is normally chosen to maximize the uniqueness of the entries in the index, while minimizing storage requirements. For Version 3c an even width (rounded up, if necessary) is always used internally.

Defaults:
i) Owner order clause: IMMATERIAL
ii) Duplicate Sorted Record Order: IMMATERIAL
iii) Sort Key Index Width: four bytes or sort key size, whichever is less

Examples:

```
ORDER IS IMMATERIAL
ORDER FIFO
LIFO
SORTED BY ASCENDING (LASTNAME,FNAME)
BY ASCENDING (LASTNAME,FNAME)
SORTED BY AZ LASTNAME ZA FNAME, DUPLICATES FIFO
ORDER IS SORTED ZA JOBCODE, DUPLICATES NOT ALLOWED, INDEX 2
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Member Section

A member section must immediately follow each owner section. The member(s) of a set is (are) described in a member section.

Required Clause

{ MEMBER IS rt-1
MEMBERS ARE (rt-2, *) }

If the set has a single member record type, its name is indicated by rt-1. If the set has multiple member record types, those record types are indicated by (rt-2, rt-3, etc.); the ordering of these multiple record type names is irrelevant. No more than 127 record types can be declared as members of a set.

Examples:

```
MEMBER IS DEPT
MEMBERS ARE (HRLYEMP, SALEMP)
MEMBERS SALEMP HRLYEMP
```

Optional Clauses

a) Set insertion clause INSERTION IS { AUTO / MANUAL }

This clause indicates whether or not the creation of a member record occurrence causes it to be automatically (AUTO) connected to an owner record.

Default: MANUAL

Examples:

```
INSERTION IS AUTO
AUTO
INSERTION MANUAL
```

b) Member order clause

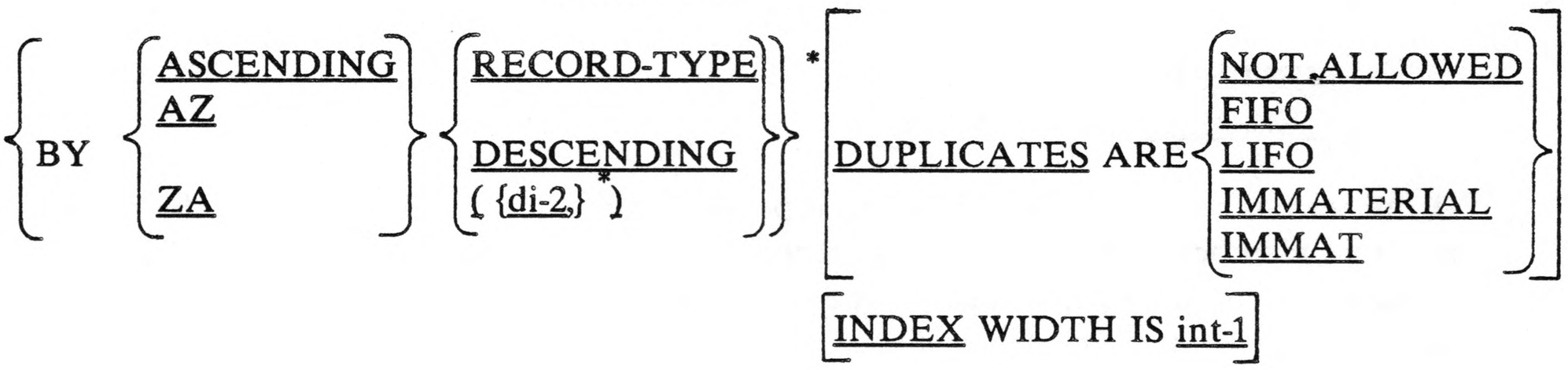
ORDER IS { FIFO / LIFO / NEXT / PRIOR / IMMATERIAL / IMMAT / SORTED <sort clause> }

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

This optional clause indicates the ordering of member record occurrences within a set occurrence. The permissible orderings are FIFO, LIFO, NEXT, PRIOR, IMMATERIAL (or IMMAT), and SORTED. The sort-clause has the same form as shown for the optional owner order clause in an owner section.

<sort-clause>:



If the member order is SORTED, then the sort- clause is required. It indicates the basis for the sorting of member record occurrences that are connected to an owner record occurrence. The sort is made on the basis of one or more data items (di-1, di-2, etc.) that have been defined for the member record type. A binary data item may not be used as a basis for sorting. If the set has multiple member record types, then the member record type names can also be used as a criterion for sorting. Thus a sort key consists of a sequence of data items, plus the possible inclusion of the literal: RECORD-TYPE. The ordering of these components within the sort key definition is from the most important to the least important. The entire sort key can be declared to be ASCENDING (or AZ) or DESCENDING (ZA). Alternatively, each component (or groups of sort key components) is declared to be either ASCENDING or DESCENDING.

Once the sort key has been specified, the designer can optionally indicate how MDBS should handle record occurrences with duplicate sort key values. The options are: NOT ALLOWED, FIFO, LIFO, IMMATERIAL (or IMMAT). The designer can also optionally indicate how many bytes (int-1) of the sort key are to be used in the multilevel member index. For instance, if int-1 is 7, then the first seven bytes of the sort key values will be used in maintaining the index. The member record occurrences are still sorted on the basis of the entire sort key. The maximum width is 28 bytes and the width is always even (rounded up, if needed) for Version 3c.

Defaults:	Member Order Clause:	IMMATERIAL
	Duplicate Sorted Record Order:	IMMATERIAL
	Sort key Index Width:	four bytes or sort key size, whichever is less

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
MDBS.DDL Specifications

Examples:

```
ORDER IS IMMATERIAL
ORDER FIFO
LIFO
SORTED BY ASCENDING (LASTNAME,FNAME)
BY ASCENDING (LASTNAME,FNAME)
SORTED BY AZ LASTNAME ZA FNAME, DUPLICATES FIFO
ORDER IS SORTED ZA JOBCODE, DUPLICATES NOT ALLOWED, INDEX 2
```

End Section

The end section terminates a DDL source specification.

Required Clause

END[.]

Example:

```
end
```

© Copyright 1987 Micro Data Base Systems, Inc.

IV. DATA SECURITY

Three approaches to data security are provided by MDBS: passwords, encryption, and access codes. The password approach is required for every MDBS data base. An application designer can optionally make use of encryption and access codes to provide very extensive data security. As discussed in Chapter II, physical security is provided by allowing some portions of the data base to be off-line while the remainder of the data base is being used.

PASSWORDS

Many data base users can be defined in the User Section of a DDL source specification. MDBS will not allow any part of the data base to be accessed through the DML in an application program or through the query system unless a user can furnish his or her assigned password. With the optional DMU module, an application developer can add, delete or change any user's password at any time, without re-initializing the data base (subject to security restrictions).

ENCRYPTION

Any data item defined in a DDL source specification can optionally be declared to be encrypted. Each occurrence of an encrypted data item is stored in an encrypted form. When an occurrence of an encrypted data item is accessed with either the DML or query language, MDBS automatically decodes the encrypted value. Thus the fact that a data item is encrypted is invisible to the users of DML application programs and the query language.

Encryption protects the data from unauthorized viewing that can occur through a sequential scan of auxiliary memory. If, for instance, employee name and salary information are stored in the same record, then a memory scan via the operating system can reveal the salary that accompanies each name. This kind of security breach is averted by declaring either (or both) the name or the salary data item to be encrypted.

ACCESS CODES

An application designer can optionally use access codes to restrict a user's access to only certain parts of a data base. MDBS differentiates between two kinds of access: read access and write access. Read access allows data to be found and viewed. Write access allows data (and data interrelationships) to be stored, modified or deleted. Table IV-1 shows what is permitted when a user has read or write access to a data item, record type, or set. This assumes that the user has furnished the correct password.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Security

Table IV-1. MDBS Access Privileges

	Read Access	Write Access
data item	Retrieve data item occurrences	Modify data item occurrences
record type	Retrieve occurrences of this record type	Create, delete, and modify occurrences of this record type
set	Locate set member records for a particular owner record	Connect member record occurrences to and disconnect set member records from a particular owner record
	Locate set owner records for a particular member record	Connect owner record occurrences to and disconnect set owner records from a particular member record

The MDBS technique for allowing the application designer to selectively restrict a user's access to various data items, record types, and sets, is based on an access code principle. In MDBS there are 16 access codes: a, b, c, ..., p. A combination of one or more access codes can be assigned to any data item to specify its read access. Another, (or the same) combination can be used to specify the data item's write access. For example, SKILCODE might have a read access of a, c, j and a write access of c-f (that is, c,d,e and f). Notice that with 16 different access codes, there are 65535 different access assignments that could be made. A combination of one or more access codes can also be assigned to any record type or set to specify its read access. Similarly, a write access combination can be specified for a record type or set.

The following rules must be observed when assigning access codes:

- A data item's read access codes must be a subset of the read access codes of its record type.
- A data item's write access codes must be a subset of the write access codes of its record type.
- A record type's read access codes must overlap (that is, intersect with) the read access codes of each set in which that record type participates.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Security

- A record type's write access codes must overlap with the write access codes of each set in which that record type participates.

An application designer can assign a combination of access codes to any user to establish read access authorization for that user. Another (or the same) combination of access codes is used to establish the write authorization for that user.

For a user to retrieve an occurrence of a data item, retrieve an occurrence of a record type, or locate a record occurrence in a set, the user read access codes must overlap the read access code of that data item, record type, or set. For example, assume a data base contains the following definitions:

- **set-1** has read access codes c, g, and n,
- **user-1** has read access codes a-d,
- **user-2** has read access codes j and p.

In this example, **user-1** has read access to **set-1** since they share a common read access code (c). **User-2** has no read access to **set-1** because he has no read access codes in common with **set-1**.

Write access codes behave identically to the read access codes. That is, if a user shares at least one write access code with the write access codes of a set, a record type, or a data item, then he may write to (modify or in some cases delete) that set, record, or data item occurrence.

No access code can be assigned to a data item, record type, or set that has not also been assigned to at least one user.

DDL Specification of Access Codes

An application designer can optionally specify a combination of access codes for any user, set, record type or data item appearing in a DDL source specification. This is accomplished by means of the optional access clause which can be:

- appended to any user specification in the user section
- included in any record type section
- included in any set section
- included in any data item section.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Security

An access clause has the same form, regardless of which section it is used in. This form is given below.

An access code list, denoted by aclist, can be either a single access code or a parenthesized list of access codes and/or access code ranges. A single access code is simply one of the letters from a through p. An access code range is indicated by two single access codes (in alphabetic order) with a hyphen between them. For instance, the access code range e-i is a shorthand way of referring to the single access codes: e,f,g,h,i.

Access clause forms:

[READ ACCESS IS aclist-1] [WRITE ACCESS IS aclist-2]

ACCESS IS aclist-3

The read access codes (for a user, set, record type, or data item) are specified in aclist-1. The write access codes are specified in aclist-2. If the read and write access codes are identical, the second form of this clause (ACCESS IS aclist-3) can be used. The codes specified in aclist-3 will be used as both the read and the write access codes.

Defaults:

i) for a user, the read and write access codes default to a

ii) for a record type, the read and write access codes default to a

iii) for a data item, the read and write access codes default to the read and write access codes of that data item's record type

iv) for a set, the read and write access codes default to a

If an application designer allows all access codes to default, then the four rules for access code assignment are satisfied. If the designer allows only some (or none) of the access codes to default, he must satisfy the assignments. Failure to observe these rules results in an error diagnostic when the DDL Analyzer processes the DDL source text.

Examples:

```
READ ACCESS IS d WRITE ACCESS IS j
READ d WRITE J
READ (a,d,m-p) WRITE (a,f-g,e,i-k)
ACCESS IS (a-e)
ACCESS (a,c,d,m-n)
READ (a,K,P)
WRITE (A-P)
```


© Copyright 1987 Micro Data Base Systems, Inc.

V. DATA INTEGRITY

With MDBS, the application developer can use either (or both) of two broad approaches to providing data integrity. One involves the inclusion of integrity checking in those DML application programs that can modify data or data relationships. The other approach is to declare various integrity constraints in the DDL source specification; these are automatically enforced by MDBS during all DML and query processing.

Incorporating integrity checking into DML application programs gives an application developer very extensive and flexible control over data integrity. For instance, compatibility checking and combination checks are straightforward within a DML application program. The degree and extent of integrity control that can be achieved within DML application programs far exceeds what can be provided with a stand-alone, nonprocedural language.

MDBS automatically enforces certain kinds of data integrity on the basis of the DDL source specification. These include the prevention of unauthorized changes, fixed set retention, exclusion of duplicate CALC key values and duplicate sort key values, and limiting the values of a data item to a desired feasible range. Of course, the integrity of 1-to-1, 1-to-many, and many-to-many relationships is automatically enforced. The DDL source specification can also be used to define a transaction logging file. All alterations to a data base since the last data base backup are recorded on this file if the MDBS-RTL package has been acquired. In the event of a breach in data base integrity, all (or, if desired, some) of the logged transactions can automatically be reapplied to the data base backup copy to reestablish the data base's integrity.

PREVENTING UNAUTHORIZED CHANGES

The MDBS data security features presented in Chapter IV give an application developer extensive and highly selective automatic control over who can make which kinds of alterations to the data and data interrelationships. This guards against both accidental and intentional assaults on data integrity.

FIXED SET RETENTION

As discussed in prior chapters, the fixed set retention option allows an application developer to declare a fixed data interrelationship. Once owner and member record occurrences are connected for a fixed set, their interrelationships can never be altered. The integrity of data relationships that should never change is, therefore, automatically maintained.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Integrity

DUPLICATE KEY EXCLUSION

If it is important to prohibit the existence of two record occurrences with the same sort key value, this can be accomplished by indicating (in a sort-clause) that duplicates are not allowed. Duplicate calc key values can also be prohibited from the data base by indicating (in a CALC-key-clause) that duplicates are not allowed.

INVALID DATES AND TIMES PROHIBITED

If a data item is declared to be of the type date, then invalid occurrences of that data item are automatically prohibited from the data base. For instance, 09/31/1957 is not a valid occurrence of a date data item. Any attempt to create such an occurrence is automatically prohibited. The maximum day allowed for a February date is 28, except in leap years. In leap years, the maximum day for a February date is 29. Not all years that are divisible by 4 are leap years. For instance, 1900 is not a leap year and such facts are known by MDBS. All leap years from the year 0000 to the year 9999 are handled by MDBS. The maximum day allowed for April, June, September or November is 30. The maximum day allowed for any other month besides February is 31.

If a time data item is declared in a DDL specification, then invalid occurrences of that data item are automatically prohibited from the data base. For instance, 221:65:03 is not a valid occurrence of a time data item. Any attempt to create such an occurrence is automatically prohibited. The maximum number of seconds that can be used in a time occurrence is 59. The maximum number of minutes that can be used in a time occurrence is 59.

FEASIBILITY RANGE SPECIFICATION

An optional clause may be used in any data item section to specify the range of permissible values for the data item being defined. Whenever an attempt is made to create or modify an occurrence of a data item, MDBS checks the new data value to determine whether it is in the range of feasible values for that data item. If it is out of the feasible range, the creation or modification does not take place and an error message to that effect is issued.

The permissible lower and upper bounds for a data item feasibility range depend on the size and type of that data item. The minimum (lower bound) and the maximum (upper bound) for each type of data item of size n is shown in Table V-1. Notice that the upper bound for date data items is 126 years above the specified lower bound. Upper and lower bounds for character and string data items must be in quotes.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Integrity

There are two feasibility range exceptions:

- binary data item types cannot use a feasibility range clause
- character data item types longer than 512 characters cannot use a feasibility range clause.

Table V-1. MDBS Data Item Feasibility Range Limits

Type	Size	Minimum Lower Bound	Maximum Upper Bound
integer	n	-2^{8n-1}	2^{8n-1}
real	n	$-2^{127}(1-256^{1-n})$	$2^{127}(1-256^{1-n})$
unsigned	n	0	$256^{n}-1$
idec	n even	$-10^{63}(1-10^{-n})$	$10^{63}(1-10^{-n})$
	n odd	$-10^{63}(1-10^{-n-1})$	$10^{63}(1-10^{-n-1})$
character	n	the lowest character in the host operating system's character code	a sequence of n characters, each of which is the highest character in the host operating systems's character code
string	n	the lowest character in the host operating system's character code	a sequence of n characters, each of which is the highest character in the host operating systems's character code
time	--	000:00:00	255:59:59
date	--	01/01/0000	126 years beyond the specified lower bound

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Integrity

The DDL Analyzer issues an error when it processes a DDL source text for any of the following conditions:

- an upper bound specification either exceeds the maximum upper bound or is less than the minimum lower bound

- a lower bound specification either exceeds the maximum upper bound or is less than the minimum lower bound

- the DDL feasibility range limits contradict each other. That is, the lower bound is greater than the upper bound.

Numeric ordering is used for numeric data items. The character code convention of the host operating system (e.g., ASCII) is used for string and character data item orderings. For string data, control characters are treated as blanks. For string and character data, some parity bits may be stripped away by MDBS. Chronological ordering is used for date and time data items.

Letting **lbound** indicate lower bound and **ubound** indicate upper bound, the optional clause for stating a data item's feasibility range is:

RANGE IS { lbound / LOWEST } TO { ubound / HIGHEST }

If **LOWEST** is declared rather than **lbound**, then the lower bound takes on a default value. If **HIGHEST** is specified rather than **ubound**, then the upper bound assumes a default value. Stating that RANGE IS **LOWEST** TO **HIGHEST** has the same logical effect as omitting the range clause; defaults are used for both the upper and lower bounds.

Defaults: With the exception of date data items, the lower bound defaults to the minimum lower bound (**LOWEST**) and the upper bound defaults to the maximum upper bound (**HIGHEST**). The lower bound default for a date data item is 01/01/1900. The upper bound default for a date data item is 126 years beyond the lower bound.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Integrity

Examples:

```
RANGE IS -273 TO 10000                    (Numeric data items)
RANGE 240 HIGHEST
RANGE 0.00 TO 4900.00

RANGE LOWEST TO 65:00:00                  (Time data item)

RANGE 01/01/1869 TO HIGHEST               (Date data item)

RANGE "A" "ZZZZZZZZ"                      (Character or String data items)
RANGE IS "AARDVARK" "ZEBRA"
RANGE "BROWN" TO "BROWN"
RANGE "3" TO "am9"
RANGE "mmm" "AXY"
```

DATA BASE RECOVERY AND TRANSACTION LOGGING

The RTL form of MDBS supports additional data integrity mechanisms including page image posting, transaction logging and transaction recovery. These mechanisms are fully described in the MDBS RTL Manual.

© Copyright 1987 Micro Data Base Systems, Inc.

VI. USING THE MDBS.DDL SOFTWARE

After designing a data base schema and specifying that schema in the MDBS Data Description Language, an application developer creates the new data base from the MDBS.DDL software and a file containing the DDL source text. MDBS.DDL software provides two functions to assist developers in creating data bases:

- DDL Analyzer - checks the syntax of DDL source texts and initializes data bases
- DDL text editor - allows you to create and edit DDL source specifications

The DDL Analyzer processes DDL source specifications for valid structure and syntax. If MDBS.DDL detects an error in the source specification the analysis stops and MDBS.DDL issues the appropriate error message. If the source specification is valid, MDBS.DDL creates the data base by initializing the data base system pages, storing the data dictionary on the system pages, and initializing the remaining (non-system) data base pages.

The DDL text editor allows you to create and edit ASCII text files for your DDL source specifications. On-line help, text entry commands, line editing commands, and global editing commands are supported with this editor; please see figure VI-1 below for the editor command levels.

MDBS.DDL operates in batch or interactive mode. Batch mode is advantageous in that it can process larger schemas than the interactive mode. In batch mode, MDBS.DDL immediately analyzes your DDL source specification without any user interaction. After successfully completing the analysis and initializing the data base, MDBS.DDL returns control to the operating system.

In interactive mode, MDBS.DDL provides you with access to the DDL text editor and DDL analyzer. This allows you to develop and refine your DDL source specification in a single environment.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

INVOKING MDBS.DDL

The procedure to invoke MDBS.DDL is operating system dependent, however, a few general guidelines exist when invoking MDBS.DDL:

- Invoke the MDBS.DDL program from the operating system by specifying the MDBS.DDL executable file name followed by the appropriate command line arguments.
- For batch mode operation, include the name of the DDL source specification file as one of the command line arguments.
- For interactive mode operation, do not specify the name of a DDL source text file.

See your MDBS System Specific Manuals for the proper syntax to invoke MDBS.DDL.

MDBS.DDL supports optional command line arguments. The arguments can appear in any order on the command line. Some arguments are valid globally (for both batch and interactive modes); other arguments are valid only for batch mode.

Command line Arguments (Global)

The following command line arguments apply to both operating modes of MDBS.DDL*.

-b*nnnn*

allocates ***nnnn*** bytes for use by MDBS.DDL. The allocation is for the number of bytes beyond the MDBS.DDL program size.

If **-b** is not used, then all available bytes are allocated to MDBS.DDL (regardless of whether or not MDBS.DDL needs all of these bytes).

-q suppresses the page initialization message during data base initialization

*The exact syntax may vary according to your operating system.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Command Line Arguments (Batch mode)

The following command line arguments apply only for the MDBS.DDL batch operating mode. Only one of these options can appear on the command line since each option specifies a different action for MDBS.DDL to take after the DDL analysis has taken place.

-s initialize a short form of the data base. This option initializes the system data base pages and stores the data dictionary on the system pages, but it does not initialize the non-system data base pages.

-o initialize an "OEM" form of the data base. This option initializes a short form of the data base (as **-s** above) and it renders the schema information inaccessible to the data base users.

-n no initialization is performed; MDBS.DDL only analyzes the source specification.

The **default initialization setting** for MDBS.DDL (no initialization option is specified) fully initializes a data base after analyzing the source specification.

INTERACTIVE MDBS.DDL USE

When used interactively, MDBS.DDL displays its banner message and the MDBS serial number on your console. Next, MDBS.DDL displays its prompt (**::**) to indicate it is ready to receive a command or DDL source text. The line you enter is either an interactive command or DDL source text. Within MDBS.DDL, you can inter-mix commands and text lines as you desire.

The input limit for a single text line in MDBS.DDL is 100 characters, including blanks. If the user attempts to enter more than 100 characters on a line, MDBS.DDL ignores the excess characters.

MDBS.DDL accepts incoming information (commands and source text) in upper or lower case. It then parses the command entered and performs it.

MDBS.DDL transposes most DDL source text to upper case before storing the schema elements in the data dictionary. Data base element names (such as data items, record types, and set names) transpose to upper case. Characters enclosed in double quotes (such as passwords, user names, and filenames) are the exception; MDBS.DDL inserts these data base elements into the data dictionary as they appear in the DDL source specification.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

MDBS.DDL allows free form input; there are no strict rules to position a DDL command or DDL source text on an input line. MDBS.DDL ignores leading blanks and tabs. For instance, the following input lines all invoke the MDBS.DDL help command:

```
::h

::    h

::        h
```

Similarly, the following input lines produce the same DDL source text (specified as line 5834 of the DDL source specification):

```
::5834 ITEM ADDRESS STRING 40

::5834      ITEM ADDRESS    STRING 40

::   5834 ITEM  ADDRESS  STRING  40
```

If the input line begins with a one to four digit numeral (0-9999), MDBS.DDL interprets the rest of the line as DDL source text. The numeral indicates the relative position of that line in the DDL source text. The other type of input line is a command, which consists of the command name followed by its appropriate arguments.

Regardless of input line type, the host operating system conventions apply for tasks such as backspacing and restarting a line. While system specific manuals cover more on this, the following general comments are usually applicable.

If MDBS.DDL encounters a command that it does not recognize (such as a misspelled command), MDBS.DDL displays the following message and prompts you for your next command:

Invalid DDL command

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Special Keys

MDBS.DDL designates several keys to perform special functions. The special keys for a given operating system depend upon the operating environment and are described in the pertinent system specific manual. The functions performed by these keys are:

- terminate an input line (RETURN)
- interrupt the line entry and restart the input line (CONTROL-X)
- backspace over and delete the previous character in the input line (CONTROL-H)
- insert a tab character in the input line (CONTROL-I)
- return control to the operating system (CONTROL-C for MS/PCDOS)
- cause the `::` prompt to appear (that is, ESCAPE for MS/PCDOS)
- toggle the DDL Analyzer output between the console and printer (that is, CONTROL-P)

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

INTERACTIVE COMMANDS

MDBS.DDL provides several interactive commands, as shown in Table VI-1. This section discusses each of these commands with its options.

Table VI-1. MDBS.DDL Interactive Commands

Help Command

h	interactive command listing and description

Text Entry Commands

d	delete one or more lines from DDL source text
i	insert one or more lines from DDL source text
l	list DDL source text
n	renumber DDL source text
r	read DDL source text from a disk file
w	write DDL source text to a disk file

Line Editing Commands

e	edit one or more lines of DDL source text

Global Editing Commands

c	search and replace (change) specific character string occurrences in the DDL source text
g	search and display (get) specific character string occurrences in the DDL source text

DDL Analysis Command

init	analyze DDL source text and initialize the data base

DDL Exit Command

stop	halt MDBS.DDL and return control to the operating system

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Help

The help command provides an alphabetic listing and a brief description of all interactive MDBS.DDL commands.

Command Syntax: **h** or **help**

Text Entry (and deletion)

A DDL source specification consists of many text lines. When a source specification enters MDBS.DDL, each line of text is assigned a number in the range of 0 through 9999. MDBS.DDL maintains these DDL source lines in numeric order.

MDBS.DDL works with one DDL source specification at any given moment. When MDBS.DDL is invoked, there is no DDL source text in use. The user either reads a file into MDBS.DDL or creates text via the MDBS.DDL text entry mode. All MDBS.DDL interactive commands (except the "read" command) operate on this local DDL source specification.

If MDBS.DDL runs out of memory while you are adding text to the source specification, the current line of text is lost and MDBS.DDL displays the following message:

Insufficient room in memory

All text entered up to this point is still preserved in the local source text. At this point you may want to save the local source specification to a disk file and maintain the DDL source specification with another text editor. Such files can be processed by the "read" and "write" commands, and by the MDBS.DDL analyzer mode.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

The following commands describe the text entry functions available in MDBS.DDL.

Text Insertion without a Command

The simplest method to create a new line of text within the local source text is to directly enter the entire source text line (preceded by a number from 0 to 9999) in MDBS.DDL. If the new line and another line of the source text have the same number, the newly entered line will replace the existing line. If there is no local text at the time of a line insertion, the inserted line becomes the local source text.

This method of inserting lines into a DDL source specification is good for small additions; however, the insert command is a better tool for extensive additions. See the insert command (next) for its capabilities.

Examples:

```
10 THIS IS LINE 10 OF TEXT
20 ANOTHER LINE OF TEXT
8393so is this
9999 HIGHEST LINE NUMBER
```

Insert Command

The insert command inserts one or more lines of text into the local source text and has two forms:

i	insert source lines beginning at the start of the local source text
i nn	insert source lines after line nn of the local source text

After you specify the insert command, MDBS.DDL prompts you for a line of source text by displaying a text line number. The text editor starts with one of three lines:

- line 0 - default case; ***nn*** was not specified in the insert command
- line ***nn*** - insert command specified line ***nn*** and it does not already exist
- line ***nn***+1 - if you specified line ***nn*** and it already exists

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Upon receiving a line number prompt, type in the line of DDL source text and press the RETURN (or ENTER) key. MDBS.DDL inserts the input line into the source specification and issues the prompt for the next line (*nn*+1, *nn*+2, and so forth). To stop the text insertion process (and return to the **::** prompt), press the RETURN key when the next line prompt appears.

Lines of existing source text following the insertion point are automatically "pushed down" and renumbered when conflicts arise.

Examples:

i	Begins insertion at line 0; where line 0 already exists, the existing lines are pushed down and renumbered.
i 2001	Begins insertions at line 2001 if that line does not already exist; insertions begin at 2002 (with pushdown renumbering) if that line already exists.

Delete Command

This command deletes one or more lines of the source specification and has three forms:

d	deletes the entire local source text
d nn	deletes line *nn* from the local source text
d nn mm	deletes lines *nn* through *mm* from the local source text

The first command form (line numbers *nn* and *mm* are not specified) clears the source text workspace; MDBS.DDL responds with the prompt of:

Delete all (y/n)

Enter *y* to clear the local workspace; enter *n* to abort the delete command.

The second and third command forms delete specific lines from the source text. *nn* and *mm* specify the source text line number(s) to delete from the source text. Commas or blanks serve as delimiters between *nn* and *mm*. MDBS.DDL deletes the source text lines and prompts you for the next DDL command (with **::**).

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Examples:

d 10	Deletes line 10
d 977,1024	Deletes lines 977 through (including) 1024
d 30 40	Deletes lines 30 through 40

d
Delete all?(y/n) MDBS.DDL prompt
y User responds with y to delete all lines.

List Command

The list command displays one or more line of text from the local DDL source text to the console and has three forms:

l	lists all lines in the local DDL source text
l *nn*	lists line ***nn*** of the local source text
l *nn mm*	lists lines ***nn*** through ***mm*** of the local source text

The first command form (line numbers ***nn*** and ***mm*** are not specified) lists the entire DDL source text on the console. The second and third forms list specific lines from the source text. ***nn*** and ***mm*** specify the source text line number(s) to list from the source text. A comma or blank can serve as the delimiter between ***nn*** and ***mm***.

When a listing completes or if there are no lines in the range to list, then MDBS.DDL prompts you for the next DDL command (with **::**).

Examples:

l	Lists all lines of text
l 0	Lists line 0
l 10 20	Lists lines 10 through 20
l22, 30	Lists lines 22 through 30
l5834,6000	Lists lines 5834 through 6000

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Renumber Command:

This command renumbers the entire source specification and has three forms:

n	renumbers all text, assigning the first line the number 10 and incrementing the line numbers by 10
n nn	renumber all text, assigning the first line the number *nn* and incrementing the line numbers by 10
n nn mm	renumber all text, assigning the first line the number *nn* and incrementing the line numbers by *mm*

In each case the renumber command renumbers the entire DDL source text. *nn* (used in the second and third command forms) specifies the number to assign to the first text line of the DDL specification. *mm* (used in the third command form) specifies the increment between consecutive lines. A comma or blank can serve as the delimiter between *nn* and *mm*.

If an increment of zero is specified or if the parameters *nn* and *mm* are such that a line number greater than 9999 would result, the message is displayed and MDBS.DDL prompts you for the next DDL command (with **::**).

requested resequencing not possible!

When the renumbering process completes, MDBS.DDL gives the **::** prompt.

Examples:

n	The first text line will be 10, the second 20, and so on
n 30	The first line will be 30, the second 40, and so on
n20, 20	The first line will be 20, the second 40, and so on
n 10 10	The first line will be 10, second 20, and so on
n0,3	The first line will be 0, the second 3, and so on

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Read Command

This command reads a previously saved text file into MDBS.DDL. The file contents becomes the local source text, replacing any existing local source text. The new local text is automatically numbered and can be operated on by any of the MDBS.DDL commands. The read command has two forms:

r	read a DDL source text (from disk), making it the local text
r *filename*	read the text file "***filename***", making it the local text

The first form of the "read" command prompts you for a file name. Enter the fully qualified file name you want to use with MDBS.DDL. If you press the RETURN (ENTER) key without a file name, MDBS.DDL returns to the : : prompt.

The second form of the read command attempts to read text into MDBS.DDL from the file "filename" without any additional prompts.

When MDBS.DDL successfully reads a file, it displays the following message indicating the size of the file transferred into the MDBS.DDL workspace and prompts you for the next DDL command (with : :).

xxxx bytes

When the source text file is larger than the MDBS.DDL workspace, MDBS.DDL displays the following message and prompts you for the next DDL command (with : :).

Insufficient room in memory

The text in memory before the "read" command is probably lost. You can still, however, initialize a data base from the oversized DDL source specification by using the "init filename" command described later in this Chapter.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

MDBS.DDL processes read errors from the operating system (caused by such things as bad disk sectors, hardware problems, an incomplete file name, and an incorrect file name) by displaying the following message:

*** * Error in reading file** ***file–1*** *** ***

If a file is not successfully read, the local text present in memory may be intact, depending on the nature of the read error.

Examples:

r DEPT.DDL	Read text from the file DEPT.DDL into MDBS.DDL
r	Read a file (unspecified) into MDBS.DDL
filename::	MDBS.DDL asks for a file name
DEPT.DDL	User responds with fully qualified file name such as DEPT.DDL

Write Command

This command allows a user to save a copy of the local source text to a disk file. The write function strips off the text line numbers; they are not stored in the disk file. Also, the write function leaves the local text copy intact. The write command has two forms:

w	write the local DDL source text onto a disk file
w *filename*	write the local DDL source text onto the "filename" disk file

The first form of the "write" prompts you for a file name. Enter the fully qualified file name you want to use with MDBS.DDL. If you press the RETURN (ENTER) key without a file name, MDBS.DDL returns to the **::** prompt.

The second form of the write command attempts to write text to the file "filename" from MDBS.DDL without any additional prompts.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

When MDBS successfully writes a file, it displays the following message indicating the size of the file transferred to disk and prompts you for the next DDL command (with : :).

xxxx bytes

MDBS.DDL processes write errors from the operating system (caused by such things as hardware malfunctions and incorrect filename) by displaying the following message:

*** * error in writing file *file–1* * ***

Examples:

w DEPT.DDL	Write an image of the local text onto the file DEPT.DDL
w	Give the "write" command without a file name
filename : :	MDBS.DDL asks for a file name
DEPT.DDL	User responds with fully qualified file name such as DEPT.DDL

Line Editing

In the line editing mode, alterations within any line of the local DDL source text can be made without retyping the entire line. The line editor is entered through the "edit" command (recall Table VI-1).

Edit Command

This command allows you to edit one or more lines of the local source specification and has three forms:

e	edit the entire local DDL source text
e *nn*	edit line ***nn*** of the local text
e *nn mm*	edit lines ***nn*** through ***mm*** of the local text

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

The first command form (line numbers *nn* and *mm* are not specified) edits the entire local source specification. The second and third command forms specify the source text line(s) you want to edit with MDBS.DDL. A comma or blank can be used as the delimiter between nn and mm.

After you specify the edit command, MDBS.DDL displays the first line to edit and displays the editor prompt]:. At this point, you can choose one of the following alternatives:

- advance to the next line of text
- change the current line of text
- exit from line editing mode

To advance to the next line of text, see Moving Through Text, the next command. To edit the text line, see the following Change Command. To exit from the edit session, see the following Stop Command.

When there is no text remaining in the range of edit lines, MDBS.DDL ends the edit session and prompts you for the next DDL command (with ::).

If an improper command syntax is entered while in the edit session, MDBS.DDL prompts you for your next edit command with a question mark (?). You are still in the edit session.

Examples:

e	Edit all text starting with the first line
e24	Edit line 24
e 10 15	Edit all text from lines 10 through 15
e 1000,1026	Edit all text from lines 1000 through 1026

Moving Through Text

You can advance from the current line of text to the next by pressing the RETURN (ENTER) key in response to the editor prompt of]: or ?. When you move to a new line of text, MDBS displays the new line of text and follows with the]: prompt. When there is no text remaining in the range of edit lines or you finish editing the last line of the local source text, MDBS.DDL ends the edit session and prompts you for the next DDL command (with ::).

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Change Command

This command allows you to replace an existing string of characters in the current line with a new string of characters. The change command has one form and operates only within an edit session.

c*d*PRESENT*d*NEW*d* where *d* is any delimiter, change the ***PRESENT*** string within the current line to the ***NEW*** replacement string.

The change command consists of four parts:

- **c** the change command
- ***d*** the delimiter
- **PRESENT** the existing string you want to replace
- **NEW** the new, replacement string

The delimiter is a single character that surrounds each string (***PRESENT*** and ***NEW***). By definition, the delimiter cannot be contained in the ***PRESENT*** string or the ***NEW*** string. The delimiter following the new is optional; you can omit it.

If the specified present string is not found, the editor responds with a ? prompt. When a change has been made, the line editor will display the revised line and will display the prompt]:.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Examples:

The examples show a sequence of changes made to the following line of source text. Also, notice the delimiter changes from one command to the next.

00010 HTIS IS TWO DEMANSTRATE THE LINE LINE EDTOR

c/HT/TH uses / as the delimiter and changes the HT to TH. This results in:

0010 THIS IS TWO DEMANSTRATE THE LINE LINE EDTOR

c.TWO.TO uses . as the delimiter and changes TWO to TO. This results in:

0010 THIS IS TO DEMANSTRATE THE LINE LINE EDTOR

cQMAQMO uses Q as the delimiter and changes MA to MO. This results in:

0010 THIS IS TO DEMONSTRATE THE LINE LINE EDTOR

c. LINE. uses . as the delimiter and deletes LINE from the line; LINE is replaced with nothing. This results in:

0010 THIS IS TO DEMONSTRATE THE LINE EDTOR

c/ED/EDI uses / as the delimiter and changes ED to EDI. This results in:

0010 THIS IS TO DEMONSTRATE THE LINE EDITOR

Repeated Changes

Within an edit session, you can make a change repeatedly, within a line and across several lines. The following form of the change command performs this function:

C makes the remembered change

Enter a *C* followed by a RETURN (or ENTER) to invoke the last remembered change.

When you enter a change, MDBS.DDL stores the change information (the PRESENT and NEW strings) internally for subsequent use. MDBS.DDL stores the information for one change only. The next change command you enter overwrites the strings (PRESENT and NEW) and becomes the new remembered change.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

When you enter an editing session, the remembered change is always empty. This state is equivalent to invoking the following command:

```
C///
```

Examples

The examples show a sequence of changes made in a single edit session to the following lines of source text.

```
0010 PRODUCT 64KGB IS NOT NEW
0020 64KGB AND 64KGB DERIVATIVES
0030 HAVE BEEN AROUND FOREVER
0040 64KGB REPORT
```

At the start of the edit session the remembered change is *C///* and MDBS.DDL displays the first line to be edited:

```
0010 PRODUCT 64KGB IS NOT NEW
```

In the example, change 64KGB to T47S. Enter the following command:

```
C.64KGB.T47S
```

This sets the remembered change to ***C.64KGB.T47S*** and MDBS.DDL changes line 10 to:

```
0010 PRODUCT T47S IS NOT NEW
```

Press the ***RETURN*** key to move to line 20. The remembered change remains ***C.64KGB.T47S*** and MDBS.DDL displays line 20:

```
0020 64KGB AND 64KGB DERIVATIVES
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Press *C* and *RETURN* to apply the remembered change. The remembered change remains *C.64KGB.T47S* and MDBS.DDL changes line 20 to:

```
0020 T47S AND 64KGB DERIVATIVES
```

Apply the remembered change again to change the second **64KGB**. This changes line 20 to:

```
0020 T47S AND T47S DERIVATIVES
```

Press the *RETURN* key to move to line 30. MDBS.DDL displays the line:

```
0030 HAVE BEEN AROUND FOREVER.
```

Press the *RETURN* key to move to line 40. MDBS.DDL displays the line:

```
0040 ... 64KGB REPORT.
```

Press *C* and *RETURN* to apply the remembered change. The remembered change remains *C.64KGB.T47S* and MDBS.DDL changes line 40 to:

```
0040 ... T47S REPORT.
```

Stop Editing Command

This command ends an edit session and MDBS.DDL prompts you for the next DDL command (with **::**). This is the form of the stop editing command.

S stop line editing

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Global Editing

In addition to line-oriented editing, MDBS.DDL provides two commands to make global changes to a DDL source specification. The get command retrieves and displays all lines containing a specified string of characters. The change command replaces all instances of a specified character string to a new character string. Both commands are discussed below.

Get and Display Command

This command searches through a source text for a specific character string and displays all lines containing this string of characters. The get command allows you to preview all lines containing a particular character string before making a global change to the string. The get command has two forms:

g/*ps*	get and display all lines containing an occurrence of the indicated string ps
g *nn mm*/*ps*	get and display all lines from ***nn*** to ***mm*** that contain ps

The first command form (line numbers nn and mm are not specified) displays all DDL source lines containing the string. The second command form restricts the search and display function to the specified range of lines. ***nn*** and ***mm*** specify the source text line numbers(s) to carry out the search and replace. As with the change command, the get command allows delimiters other than the / symbol.

Examples:

g/retention	displays all lines containing "retention"
g 25 100/RETENTION	displays all lines in the range 25 through 100 that contain "RETENTION"
g.nodup	displays all lines containing "nodup"
g 16,83.FIFO	displays all lines in the range 16 through 83 that contain "FIFO"

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Change Command:

This command searches through a source text for a specific character string and replaces all occurrences of this string with a new string of characters. The change command has three forms:

c/*ps*/*ns*	change all occurrences of the PRESENT string to the NEW string
c *nn*/*ps*/*ns*	change occurrences of the PRESENT string in line nn to the NEW string
c *nn mm*/PRESENT/NEW	change occurrences of the PRESENT string in lines nn through mm to the NEW string

The first command form (line numbers nn and mm are not specified) changes all occurrences of the PRESENT string to the NEW string throughout the entire source text. The second and third forms restrict the search and replace functions to the specific lines. ***nn*** and ***mm*** specify the range of source text line(s) to carry out the search and replace. As with the change command, the get command allows delimiters other than the / symbol.

Examples:

c/EMPLOYEE/emp	changes all instances of "EMPLOYEE" to "emp"
c/nodup/	changes each instance of "nodup" to a blank
c 75 90/FIFO/LIFO	changes all instances of "FIFO" in lines 75 through 90 to "LIFO"
c 1,23!STR 10!STR 12	changes all "STR 10" string in lines 1 through 23 to "STR 12" (uses an exclamation mark as a delimiter)

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

DDL Analysis

The DDL analysis mode allows you to check the syntactic and logical consistency of a DDL source text specification and initialize a new data base from the specification. The init (or ddl) command has two forms:

init	Analyze and initialize the local DDL source specification
ddl	same as ***init*** above
init *filename*	Analyze and initialize the DDL source specification on the indicated disk file

The first form of this command uses the local DDL source specification. The second command form uses the DDL source specification from the named file.

The DDL analyzer systematically examines each section of the DDL source specification for proper syntax and logical consistency. If the DDL Analyzer detects no errors in the DDL source text, MDBS.DDL displays the following messages and gives you the option to initialize a data base for that DDL specification.

***** Schema description successfully analyzed *****

Do you wish to initialize the data base? (y/n/s/o)

If the DDL Analyzer detects an error in the DDL source text, MDBS.DDL does three things:

- stops analyzing the source specification at the point of the error
- displays a message indicating the nature of that error
- prompts you for the next DDL command (with : :)

Errors that occur during DDL analysis are catalogued in Chapter VII. Each error listing includes a description of the error condition along with possible causes of the error.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Initializing a Data Base

When a DDL source specification successfully passes the DDL analysis, MDBS.DDL gives you the option to initialize a data base with that schema definition stored in the data base data dictionary. MDBS.DDL displays the following message and lets you select one of four choices:

Do you wish to initialize the data base? (y/n/s/o)

If you select ***y***, MDBS.DDL completely initializes the data base. This causes MDBS.DDL to do the following actions:

- initializes the system pages in each data base area
- stores the data dictionary in the system pages of the main data base area
- initializes all non-system pages in each data base area

If you select ***n***, then MDBS.DDL does not initialize a data base and MDBS.DDL prompts you for the next DDL command (with `::`).

If you select ***s***, MDBS.DDL initializes a short form of the data base. This option is similar to the complete initialization above, but MDBS.DDL does not initialize any non-system (data) pages of the data base. The short initialization performs the following functions:

- initializes the system pages in each data base area
- stores the data dictionary in the system pages of the main data base area
- leaves all non-system data base pages uninitialized

The s option keeps the size of each data base area compact and makes it easy to install the new data base on one or more machines; under normal circumstances, you can fit several short (unexpanded) data base areas on a single floppy disk.

Once the data base is on the target hardware (disk), you can expand each data base area to an appropriate size for the application with the optional DMU module. You must expand a data base that has been initialized with the short option because the data base control system cannot store information on unexpanded data base pages.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

The *o* option initializes an OEM (original equipment manufacturer) form of the data base. This option performs a short initialization of the data base and additionally it prevents data base users and other developers from seeing the data base schema. The OEM initialization includes the following functions:

- initializes the system pages in each data base area
- stores the data dictionary in the system pages of the main data base area
- leaves all non-system data base pages uninitialized
- disables schema viewing functions and commands

After initializing the data base with the short, OEM, or complete initialization (and accompanying transaction log file where it is specified), MDBS.DDL displays the following data base statistics to the console:

- the page size of each data base area
- the number of pages allocated in each area
- the number of system pages of each area
- the remaining bytes of each area
- the minimum single user DMS buffer size in bytes

The single user DMS buffer size is the smallest amount of contiguous memory MDBS.DMS needs to transfer information from the data base control system to your application program. This size (in bytes) is exclusive of any file control or data blocks that may be needed by a host language program. In some host programming languages, you can control the buffer size with the SETPBF command as described in the MDBS III DMS Manual. Chapter VII of your MDBS III System Specific Manual indicates if SETPBF is available for your host programming language.

After MDBS.DDL initializes the data base and displays the statistics, it prompts you for the next DDL command (with `::`).

Examples:

```
init
ddl DEPT.DDL
init DEPT.DDL
```


© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Using DDL Software

Stop COMMAND

The stop command halts MDBS.DDL and returns control to the operating system. The local DDL source text is destroyed (unless you have previously saved it with the write command.

stop or **bye** or **end** or **quit**

© Copyright 1987 Micro Data Base Systems, Inc.

VII. DDL ANALYZER MESSAGES

This chapter describes the error messages that can result from the DDL Analyzer. These errors arise from improper syntax or errors in the logical structure specified in the DDL description. Each diagnostic message consists of the following information and presents it in the accompanying layout:

***** This line states the error message MDBS.DDL reports.**

Explanation:
The explanation is a synopsis of the error; it describes the condition identified by the DDL analyzer.
Possible Causes and/or Solutions:
1. These items help you to locate the portion of the specification that produced the error.
2. They also provide corrective action you can take to remedy the problem.

ERROR MESSAGES

***** A CALC key item has not been defined.**

Explanation:
All items forming part of the CALC key for a record type must be defined as data items for that record type.
Possible Causes and/or Solutions:
1. One or more data items forming part of the CALC key for a record type have not been defined.
2. Check data item specifications in RECORD and ITEM sections.

***** Access level not in the range a to p.**

Explanation:
The access level defined for the construct (item, record, or set) is not in the range 'a' to 'p'.
Possible Causes and/or Solutions:
1. The access level for any item, record, or set must be in the range 'a' to 'p'.
2. Check item, record, and set access level descriptions.

***** An item in the sort clause is not in a record.**

Explanation:
An item declared in the sort clause has not been defined in the record type.
Possible Causes and/or Solutions:
1. A set has been declared sorted by an item which has not been defined earlier in the owner/member record type.
2. Check item definitions in owner/member record types.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Area and data base page sizes must be equal.**

Explanation:

The page size defined for the data base and area(s) must be equal. This error occurs only in Version 3a.

Possible Causes and/or Solutions:

1. Check page size definitions in the IDENTIFICATION and AREA sections.

***** Area not previously defined.**

Explanation:

The area has not been defined earlier.

Possible Cause and/or Solutions:

1. Check the main area definition in IDENTIFICATION section, and subsequent area specifications in the AREA section.

***** Area page exceeds data base page size.**

Explanation:

The page size for one or more areas is greater than the page size defined for the main data base area. This error occurs only in Version 3c.

Possible Causes and/or Solutions:

1. The page size for any of the areas must be less than or equal to the page size defined for the main data base area.
2. Check page size definitions for the various areas.

***** AREA or RECORD section missing or misplaced.**

Explanation:

The system was expecting an AREA or RECORD section based on an end-of-section descriptor.

Possible Causes and/or Solutions:

1. Check end-of-section descriptors (e.g., a period) in the AREA and RECORD sections.
2. Check AREA and RECORD section specifications.

***** A sort—key item is not encrypted across participating records.**

Explanation:

For sets with multiple owner/member record types, the sort-key item is not encrypted across participating owner/member record types; i.e., the item has been defined to be encrypted in one or more owner/member record types, and not encrypted in one or more owner/member record types.

Possible Causes and/or Solutions:

1. For sets sorted by their multiple owner/member record types, the sort key item, if defined to be encrypted in any one owner/member record type, must be defined as encrypted in all participating owner/member record types.
2. Check sort-key item description in participating owner/member record types.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** A sort—key item length not constant across participating records.**

Explanation:

For sets with multiple owner/member record types, the length of the sort-key item is not constant across participating owner/member record types. This error occurs only for sort-key items which have been declared as string items.

Possible Causes and/or Solutions:

1. For sets sorted by their multiple owner/member record types, the length of the sort key item must be the same in the participating owner/member record types.
2. Check sort-key item description in participating owner/member record types.

*****A sort—key item replication count not equal across participating records.**

Explanation:

For sets with multiple owner/member record types, the sort-key item replication count is not equal across participating owner/member record types.

Possible Causes and/or Solutions:

1. For sets sorted by their multiple owner/member record types, the replication count for the repeating sort-key item must be the same in the participating owner/member record types.
2. Check sort-key item description in participating owner/member record types.

***** A sort—key item type not equal across participating records.**

Explanation:

For sets with multiple owner/member record types, the sort key item type is not comparable across participating owner/member record types.

Possible Causes and/or Solutions:

1. For sets sorted by their multiple owner/member record types, the sort key item type must be the same in the participating owner/member record types (e.g., CHAR, REAL).
2. Check sort-key item description in participating owner/member record types.

***** At least one owner is required in a set.**

Explanation:

A set description requires at least one owner record type.

Possible Causes and/or Solutions:

1. Check 'owner' clause description of SET section.

***** At least one member is required in a set.**

Explanation:

A set description requires at least one member record type.

Possible Causes and/or Solutions:

1. Check 'member' clause description of SET section.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Bad character in number field.**

Explanation:
A number field in a RANGE specification has a bad character.
Possible Causes and/or Solutions:
1. Check RANGE specification for this item.

***** Binary items cannot be ranged.**

Explanation:
A range cannot be specified for binary items.
Possible Causes and/or Solutions:
1. Check binary item descriptions in ITEM section.

***** Binary items cannot be part of a sort—key.**

Explanation:
A sort-key item cannot include a binary item.
Possible Causes and/or Solutions:
1. Check item descriptions of sort-key items.

***** Cannot open file.**

Explanation:
An error has occurred while trying to open a file for initializing the data base.
Possible Causes and/or Solutions:
1. Disk on which file is to be opened may be full.
2. Bad sector on disk.

***** Data base IDENTIFICATION section missing or out of order.**

Explanation:
Self-explanatory.
Possible Causes and/or Solutions:
1. Check sequence of different sections in DDL description.

***** Data base must have at least one record type.**

Explanation:
Self-explanatory.
Possible Causes and/or Solutions:
1. The DDL specification contains no record type declaration.
2. The record type declarations are not in the proper position (e.g., they follow the set declarations).

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Duplicate area name.**

Explanation:
The same area name has been defined twice.
Possible Causes and/or Solutions:
1. Check area descriptions.

***** Duplicate area specified in area list.**

Explanation:
An area name has been specified twice in the area list of a record description.
Possible Causes and/or Solutions:
1. Check area specifications in the RECORD section.

***** Duplicate clause has an unrecognizable descriptor.**

Explanation:
The duplicate clause in the RECORD section does not have a valid descriptor.
Possible Causes and/or Solutions:
1. The duplicate clause in the RECORD section does not have any of the following descriptors: NOT ALLOWED, LIFO, FIFO, IMMAT.
2. Check duplicate clause section in RECORD section.

***** Duplicate item within this record.**

Explanation:
A record type has a duplicate item specification.
Possible Causes and/or Solutions:
1. Check item specifications for different record types.

***** Duplicate member record types for this set.**

Explanation:
A set has the same record type specified twice as a member.
Possible Causes and/or Solutions:
1. Check member clause specification in SET section.

***** Duplicate owner record types for this set.**

Explanation:
A set has the same record type specified twice as an owner.
Possible Causes and/or Solutions:
1. Check owner clause specification in SET section.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Duplicate user.**

Explanation:
A user has been defined twice with different (or same) passwords.
Possible Causes and/or Solutions:
1. Check user-password specifications in USER section.

***** Duplicate record type name.**

Explanation:
The same record type has been specified twice.
Possible Causes and/or Solutions:
1. Check record type names in RECORD section.

***** Duplicate set name.**

Explanation:
The same set name has been specified twice.
Possible Causes and/or Solutions:
1. Check set name specifications in SET section.

***** Excessive date or time field.**

Explanation:
The range specified for the date or time field is excessive.
Possible Causes and/or Solutions:
1. Check possible ranges for date and time fields.
2. Check range specifications for date and time fields.

***** Excessive width for sort—key.**

Explanation:
The width specified for a sort-key exceeds the size of the sort- key's fields.
Possible Causes and/or Solutions:
1. Add the sizes of the sort-key's fields to determine the maximum sort-key width.

***** Expecting a number.**

Explanation:
A character other than a number was specified where the system was expecting a number.
Possible Causes and/or Solutions:
1. Check required specification of construct.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Expecting a quoted file name.**

Explanation:
A file name was not specified in quotes.
Possible Causes and/or Solutions:
1. Any file name in the DDL description should be enclosed in quotes.
2. Check file name specifications in IDENTIFICATION and AREA sections.

***** Expecting SET or END statement.**

Explanation:
The system was expecting a SET or END statement based on an end- of-section descriptor.
Possible Causes and/or Solutions:
1. Check end-of-section descriptors (e.g., a period) in the SET section.
2. Check SET and END sections.

***** Expecting 'RECORD—TYPE' statement in sort—clause.**

Explanation:
The system was expecting a 'RECORD-TYPE' statement in the sort- clause of the SET section.
Possible Causes and/or Solutions:
1. Check sort clause specification of SET section.

***** High date must not precede low date.**

Explanation:
In the range specification for a date item type, the high date must be specified after the low date.
Possible Causes and/or Solutions:
1. The high date has been specified before the low date in the range specification for a date item type.
2. Check range specifications in date item type.

***** Inconsistent insertion for "set—name"**

Explanation:
A set's insertion mode(s) must be consistent with set type and record location.
Possible Causes and/or Solutions:
1. Both owner and member insertion are automatic.
2. Member (owner) insertion is automatic with member (owner) records being placed in the area of the owner (member).
3. Members are declared to be clustered in an owner area and vice versa; or, clustering is requested for a recursive set.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Inconsistent item and record read access codes.**

Explanation:
The read access codes defined for any item must be a subset of the read access codes for the record type containing that item.
Possible Causes and/or Solutions:
1. The read access codes defined for the item and the record type containing that item have no elements in common.
2. Check item and record type read access code specification.

***** Inconsistent item and record write access codes.**

Explanation:
The write access codes defined for any item must be a subset of the write access codes for the record type containing that item.
Possible Causes and/or Solutions:
1. The write access codes for the item and the record type containing that item have no elements in common.
2. Check item and record type write access code specifications.

***** Inconsistent set and record read access codes.**

Explanation:
The read access codes defined for the record type and the set having the given record type as owner/member must have at least one element in common.
Possible Causes and/or Solutions:
1. The read access codes for the record type and the set having that record type as owner/member have no elements in common.
2. Check record type and set read access code specifications.

***** Inconsistent set and record write access codes.**

Explanation:
The write access codes defined for the record type and the set having the given record type as owner/member must have at least one element in common.
Possible Causes and/or Solutions:
1. The write access codes for the record type and the set having that record type as owner/member have no elements in common.
2. Check record type and set write access code specifications.

***** Insufficient room for DDL processing.**

Explanation:
There is insufficient room in memory for building tables during DDL processing.
Possible Causes and/or Solutions:
1. There is not enough room in memory for the DDL analyzer to build tables.
2. If the file has been read into memory using the 'R' command, the 'INIT' command may instead be used to initialize the data base without reading the file into memory.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Insufficient room in memory.**

Explanation:
There is not enough room in memory for any further processing, (i.e., to read a text file, to input a line of DDL source specification, etc.).
Possible Causes and/or Solutions:
1. Not enough room is available to read a file into memory.
2. Not enough room is available to input a line of DDL text.
3. The 'INIT' command may be used instead of a 'R' command to initialize a data base without reading the file into memory.
4. A text editor may be used instead of the DDL to input DDL source specification.

***** Integers may not have fractional parts.**

Explanation:
The range specifications for an item defined as an integer have been declared incorrectly.
Possible Causes and/or Solutions:
1. An integer item has a fractional part in its range specifications.
2. Check range specifications of item.

***** Invalid date or time delimiter.**

Explanation:
An invalid delimiter has been specified in the range clause for a date or time field.
Possible Causes and/or Solutions:
1. A character other than a '/' (for a date field), or other than a ':' (for a time field) has been specified as a delimiter on the range clause for these items.
2. Check range clause specifications for date and time data items.

***** Item length must be at least 1.**

Explanation:
The minimum size (number of bytes) for a data item, except a date or time item, is 1.
Possible Causes and/or Solutions:
1. The size specified for any item (except date and time data items) must be at least 1.
2. Check size specifications in data item section.

***** Item or set section missing or misplaced.**

Explanation:
The system was expecting an ITEM or SET statement based on an end-of-section descriptor.
Possible Causes and/or Solutions:
1. Check end-of-section descriptors (e.g., a period) in the ITEM and SET sections.
2. Check ITEM and SET section specifications.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Item access codes must be a subset of the record.**

Explanation:
The access codes defined for any item must be a subset of the access codes for the record type containing that item.
Possible Causes and/or Solutions:
1. An access code has been defined for the item which is not defined as an access code for the record type containing that item.
2. Check item and record type access code specifications.

***** Item length too long.**

Explanation:
The length of the data item is larger than the maximum allowable for that data item type.
Possible Causes and/or Solutions:
1. The item size (number of bytes) is longer than the maximum permissible for that data item type.
2. Check maximum allowable lengths (sizes) for different data item types.
3. Check item size specifications.

***** Item type was not specified.**

Explanation:
The type of the data item (e.g., CHAR,BIN) was not specified in the DDL source specification.
Possible Causes and/or Solutions:
1. The data item specification did not include the type of the data item (e.g., CHAR,BIN,IDEC).
2. Check data item specifications.

***** Maximum of 7 decimal positions for IDEC fields.**

Explanation:
For IDEC fields, the number of digits to the right of the decimal point cannot be greater than 7.
Possible Causes and/or Solutions:
1. A number greater than 7 has been specified for the number of digits to the right of the decimal position in the size specification for an IDEC item.
2. Check item size specification of IDEC items.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Missing set with record clustering for "record—type—name".**

Explanation:
The set specified in the record clustering clause for the indicated record type is missing from the SET section of the DDL specification.
Possible Causes and/or Solutions:
1. The set specified in the clustering clause has not been defined in the SET section.
2. Typographical error in specifying set name.
3. Check SET section and clustering specifications in RECORD TYPE section.

***** Misspecified access code range.**

Explanation:
The range for the access code has not been specified correctly. (e.g., f-a, etc.)
Possible Causes and/or Solutions:
1. An error has been made while specifying the range of access codes for the construct.
2. Check access code specifications.

***** Multiple owner/member declaration incorrect.**

Explanation:
The specification of multiple owners/members for a set is incorrect.
Possible Causes and/or Solutions:
1. While specifying multiple owners/members for a set, the format 'OWNER/MEMBER IS' rather than 'OWNERS/MEMBERS ARE' has been used.
2. Check owner/member specifications.

***** Need at least one ascending/descending clause.**

Explanation:
For a set declared to be sorted, at least one ascending/descending clause has to be specified in the sort clause.
Possible Causes and/or Solutions:
1. A set has been declared to be sorted, and no ascending/descending clause has been specified as part of the sort clause in the SET section.
2. Check sort clause specifications in SET section.

***** No item list in key—clause.**

Explanation:
The calc key clause (record location clause) in the RECORD section has no item list specified.
Possible Causes and/or Solutions:
1. No items have been specified in the calc key clause (or record location clause) in the RECORD section of the DDL specifications.
2. Check record location clause in RECORD section.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Not enough pages allocated.**

Explanation:

Not enough pages have been allocated for the main area and/or other area(s) to initialize the data base.

Possible Causes and/or Solutions:

1. The number of pages allocated for the main area and/or other area(s) is not sufficient to hold the system tables.
2. Increase the number of pages allocated for the main area and/or other area(s).

***** No user can access construct.**

Explanation:

An access code specification has been set up such that the construct (item, record, or set) cannot be accessed by any user.

Possible Causes and/or Solutions:

1. The access code specifications for this construct do not overlap (have no code in common) with the access codes of any of the users. No user can, therefore, access this construct.
2. Check access code specifications for this construct and user access code specifications.

***** Numerical overflow.**

Explanation:

This error arises from improper range specifications for a data item type. Self-explanatory.

Possible Causes and/or Solutions:

1. The range specification for a data item type has caused a numerical overflow.
2. Check range specifications in ITEM section.

***** Numerical underflow.**

Explanation:

This error arises from improper range specifications for a data item type. Self-explanatory.

Possible Causes and/or Solutions:

1. The range specification for a data item type has caused a numerical underflow.
2. Check range specifications in ITEM section.

***** Only one title allowed per construct.**

Explanation:

Any area, item, record type, or set can have at most, one title specification.

Possible Causes and/or Solutions:

1. More than one title has been specified for a construct (area, item, record type or set).
2. Check title clause specifications for the construct.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Page size must be a multiple of 256 bytes.**

Explanation:

This error arises only in Version 3a. The page size (for the main area and area(s)) must be a multiple of 256 bytes.

Possible Causes and/or Solutions:

1. The page size defined for the main area and/or other area(s) is not a multiple of 256 bytes.
2. Check page size specification in IDENTIFICATION and AREA sections.

***** Page size must be at least 256 bytes.**

Explanation:

A page size must be at least 256 bytes.

Possible Causes and/or Solutions:

1. The page size defined for an area is less than 256 bytes.
2. Check page sizes specification in IDENTIFICATION and AREA sections.

***** Reals must be at least 2 bytes long.**

Explanation:

The minimum item size specification for a REAL data item is 2 bytes.

Possible Causes and/or Solutions:

1. The item size specification for a data item defined as REAL is less than 2 bytes.
2. Check item size specifications.

***** Record name not present with clustering for "set—name"**

Explanation:

The record type specified does not properly participate in the SET name indicated in the clustering clause.

Possible Causes and/or Solutions:

1. A record type having a clustering clause has not been defined as the owner/member of the set indicated in the clustering clause.
2. Typographical error in record type specification.
3. Check owner/member specifications and record type specification in clustering declaration.

***** Record too big —— increase page size.**

Explanation:

The page size specified must be at least as large as the length of the largest record type defined in the data base.

Possible Causes and/or Solutions:

1. The page size is less than the length of the largest record type defined in the data base.
2. For the same length of the above record type, the page size must be increased.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Record type not previously defined**

Explanation:

A record type which has not been defined earlier has been used in the specification of a section of DDL text.

Possible Causes and/or Solutions:

1. A record type used in a particular section of DDL text has not been defined earlier in the RECORD section.
2. Typographical error in record type specification.
3. Check record type specification and subsequent usage.

***** Replication already specified.**

Explanation:

The replication factor in the 'occurs' clause of the ITEM section has already been specified.

Possible Causes and/or Solutions:

1. The 'occurs' clause of the ITEM section has been specified twice.
2. Check 'occurs' clause in ITEM section.

***** Requested resequencing not possible using REN.**

Explanation:

An improper numbering sequence has been specified, or the numbering sequence specified extends beyond line 9999.

Possible Causes and/or Solutions:

1. The numbering sequence specified causes the file to extend beyond line 9999.
2. A negative increment (decrement) has been specified using the REN command.
3. Respecify REN command.

***** Sorting a single record—type is foolish.**

Explanation:

The set order clause is sorted by RECORD-TYPE and only one record type participates as the set owner or member. Sorting by record type implies that at least two record types participate as the set owner or member.

Possible Causes and/or Solutions:

1. Check the set order clause(s) of this set.
2. Check the set owner and member record types. To use RECORD-TYPE in a sort clause, you must have multiple owner or member record types in the set.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** SYSTEM cannot be a member of a set.**

Explanation:
The SYSTEM record type can only be declared as an owner, and not as a member of any set.
Possible Causes and/or Solutions:
1. SYSTEM has been declared as a member of some set.
2. Check owner/member specifications.

***** SYSTEM cannot own a multiple owner set.**

Explanation:
A set with multiple owner record types cannot have SYSTEM as an owner.
Possible Causes and/or Solutions:
1. Check the set's owner clause.

***** Text had to be renumbered.**

Explanation:
The text of the DDL specification had to be extended beyond line 9999.
Possible Causes and/or Solutions:
1. The numbering sequence specified in the DDL command would have caused the file to extend beyond line 9999.
2. Respecify DDL command.

***** The decimal point position exceeds the size.**

Explanation:
The position specified for the decimal point in internal decimal (IDEC) fields cannot be greater than the size of the field.
Possible Causes and/or Solutions:
1. The decimal point position on an IDEC field is larger than the size (number of bytes) specified for the data item.
2. Decrease decimal point position so that it is less than the item size.
3. Increase the item size (number of bytes).

***** Too many areas.**

Explanation:
The number of areas specified in the AREA section is more than the maximum number of permissible areas.
Possible Causes and/or Solutions:
1. More than 15 areas (aside from the main data base area) have been specified in the AREA section for an environment that allows up to 16 areas per data base.
2. More than 7 areas (aside from the main data base area) have been specified in the AREA section for an environment that allows up to 8 areas per data base.
3. Check area specifications in AREA section.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Too many owners/member in set.**

Explanation:
A set can have at most 127 record types defined as owner/members.
Possible Causes and/or Solutions:
1. More than 127 record types have been defined as owners/members for the set.
2. Check owner/member specifications for set.

***** Too many pages per area.**

Explanation:
Too many pages are specified for an area.
Possible Causes and/or Solutions:
1. The pages specified for an area exceeds 4095 (for environments that allow 16 areas per data base) or 8191 (for environments allowing no more than 8 areas per data base).

***** Type already specified.**

Explanation:
A type specification for a data item has already been made.
Possible Causes and/or Solutions:
1. The type specification for a data item is made twice (e.g., CHAR 10 BIN).
2. Check item specifications in ITEM section.

***** Unable to create or open file.**

Explanation:
The system has encountered an error when trying to open or create a main area or other area file.
Possible Causes and/or Solutions:
1. Not enough room on disk.
2. Bad sector on disk.

***** Unrecognizable identifier.**

Explanation:
The system was expecting a valid identifier name.
Possible Causes and/or Solutions:
1. The name for a data base, area, set, record type or data item exceeds eight characters.
2. The name for a data base, area, set, record type or data item contains non-alphanumeric characters.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Analyzer Messages

***** Unrecognizable or misplaced input.**

Explanation:

The indicated DDL text input is out of sequence, has a missing word, or contains a misspelled term.

Possible Causes and/or Solutions:

1. Check sequence of DDL text input.
2. Check syntax of DDL text input.
3. Check keywords specification in DDL text input.
4. Check the set order clause in the DDL text input. MDBS.DDL allows one set order clause to accompany each set owner and each set member clause.

***** Upper value in range is less than lower value.**

Explanation:

Self-explanatory.

Possible Causes and/or Solutions:

1. Check range specifications in ITEM section.

***** User section missing or misplaced.**

Explanation:

The system was expecting a USER statement based on an end-of- section descriptor.

Possible Causes and/or Solutions:

1. Check end-of-section descriptor (e.g., a period) in USER section.
2. Check user-password specifications in USER section.

***** Write error.**

Explanation:

The system has encountered an error while trying to write to disk during the data base initialization procedure.

Possible Causes and/or Solutions:

1. The number of pages defined for the main data base is less than the number of system pages required to initialize the data base.
2. Increase the number of pages specified in IDENTIFICATION section.

© Copyright 1987 Micro Data Base Systems, Inc.

APPENDIX A

DATA DESCRIPTION LANGUAGE KEYWORDS

MDBS DDL MANUAL
Keywords

APPENDIX A

MDBS.DDL recognizes the following as keywords:

access
allowed
any
are
area
ascending
auto
az
bin
binary
by
bytes
calc
char
character
database
date
db
decimal
default
descending
duplicates
encrypted
end
fifo
file
fixed
for
highest
idec
immat
immaterial
in
index
insertion
int
integer
is
item
key
keys
language
lifo
log
logfile
lowest
manual
member
members
name
next
nodup
not
occurs
of
optional
order
owner
owners
page
pages
prior
range
read
real
record
retention
set
size
sorted
str
string
syn
synonym
time
times
title
to
type
unsigned
user
width
with
within
words
write
za

No keyword can be used as an identifier, unquoted password, or unquoted user name in DDL source specification unless it is preceded by one of the following words: IS, ARE, WITH, OF, TO, or BY.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

APPENDIX B

MDBS DATA DESCRIPTION LANGUAGE SYNTAX

MDBS DDL MANUAL
Data Description Language Syntax

APPENDIX B

See Chapters III-V for a detailed presentation of the DDL.

NOTATION:

<u>underlined</u>	An underlined expression must appear in the clause.
[ ]	Zero or one of the alternatives within the brackets can appear in the clause.
{ }	One of the alternatives within the braces must appear in the clause.
{ }*	One or more of the alternatives within the braces can appear in the clause.

DEFINITIONS:

<access-clause>:

```
{[READ ACCESS IS aclist=1] [WRITE ACCESS IS aclist=2]}
{ACCESS IS aclist=3       }
```

<CALC-key-clause>:

```
{CALC KEY } [IS ]                 ( {id=3,}* ) [NODUP ALLOWED                    ]
{CALC KEYS} [ARE]                              [DUPLICATES ARE [NOT] ALLOWED     ]
```

<sort-clause>:

```
{BY {ASCENDING } {RECORD-TYPE   } }* [DUPLICATES ARE {NOT ALLOWED}]
    {AZ        } {( {di=2,}* )  } }                  {FIFO       }
    {DESCENDING}                                     {IMMATERIAL }
    {ZA        }                                     {IMMAT      }

[INDEX WIDTH IS int=1]
```

<comment>:

```
/* comments */
```

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Description Language Syntax

ORDERING OF DDL SECTIONS:

Identification Section
User Section
Area Section (optional)
Record Sections
    Data Item Sections for each record section
Set Sections
    Owner Section for each set section
    Member Section for each set section
End Section

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Description Language Syntax

Identification Section

{DATABASE / DB} NAME IS id-1

[FILE NAME IS "file-1"]

[SIZE IS int-1 PAGES]

[PAGE SIZE IS int-2 [BYTES / WORDS]]

[{LOG FILE / LOGFILE} NAME IS "file-2"]

[TITLE IS "string-1"]

[{{SYN / SYNONYM} IS id-2}*]

DEFAULT SIZE FOR {UNSIGNED / REAL / INTEGER / INT / BINARY / BIN / CHARACTER / CHAR / STRING / STR / DECIMAL / IDECx} {int-3 / int-4[,int-5]} *

LANGUAGE IS {DANISH / ENGLISH / FINNISH / FRENCH / GERMAN / NORWEGIAN / SWEDISH}

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Description Language Syntax

User Section

USER IS { usr / "usr" } [WITH [,] { pass / "pass" }]

[<access-clause>]

Area Section (zero, one, or more per schema)

AREA NAME IS id-1

[FILE NAME IS "file-1"]

[SIZE IS int-1 PAGES]

[PAGE SIZE IS int-2 [BYTES / WORDS]]

[POINTERS ARE [NOT] ALLOWED]

[TITLE IS "string-1"]

[{ { SYN / SYNONYM } IS id-2 }*]

Record Section (one or more per schema)

RECORD NAME IS id-1

[WITHIN / IN { { ANY AREA / ({ ar-1, }*) } [<CALC-key-clause>] / AREA OF { OWNER / MEMBER } OF id-2 }]

[<access-clause>]

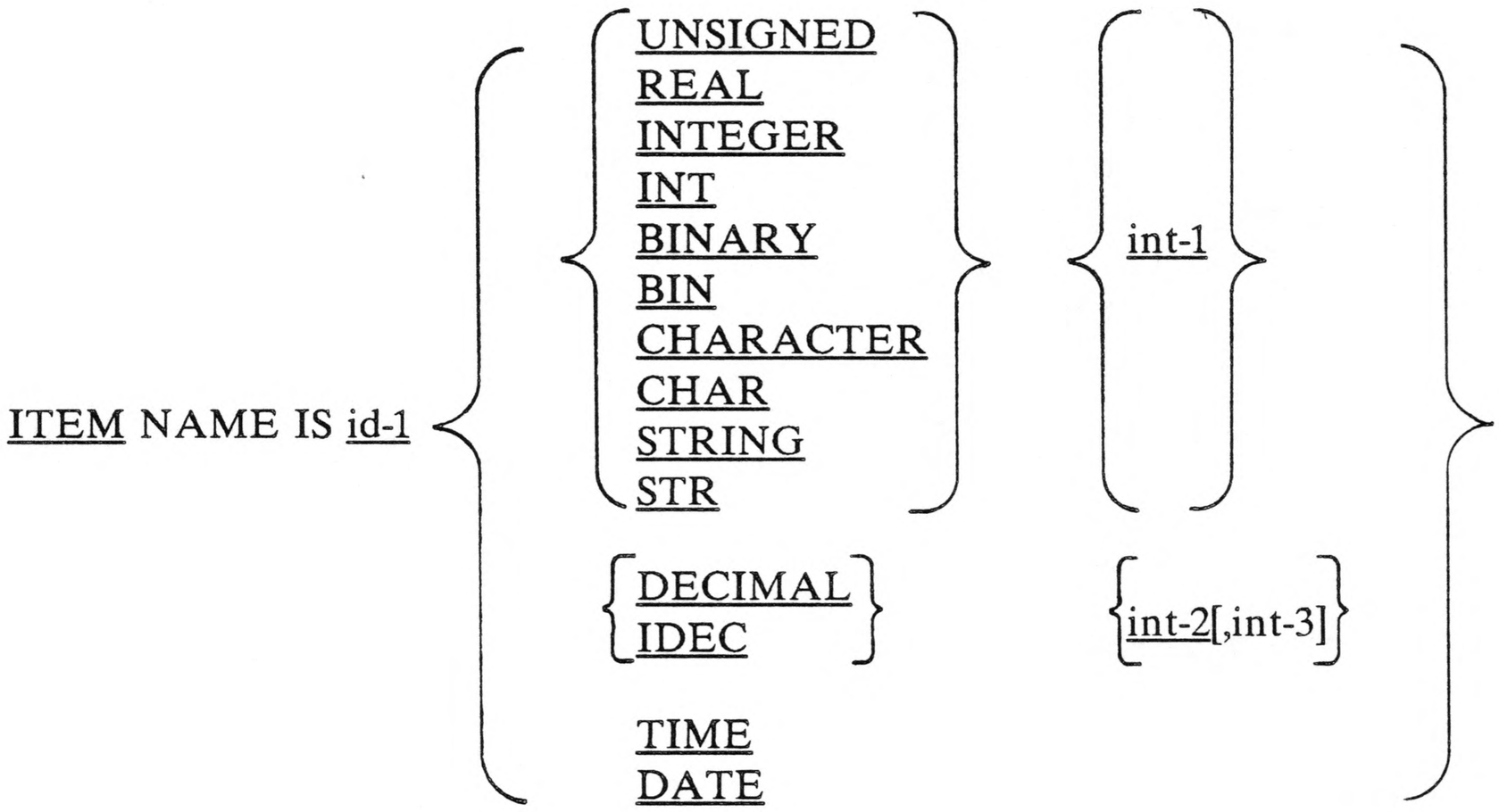
[TITLE IS "string-1"]

[{ { SYN / SYNONYM } IS id-2 }*]

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Description Language Syntax

Item Section (zero, one, or more per record type)



[OCCURS int-4 TIMES]

[IS ENCRYPTED]

[<access-clause>]

[TITLE IS "string-1"]

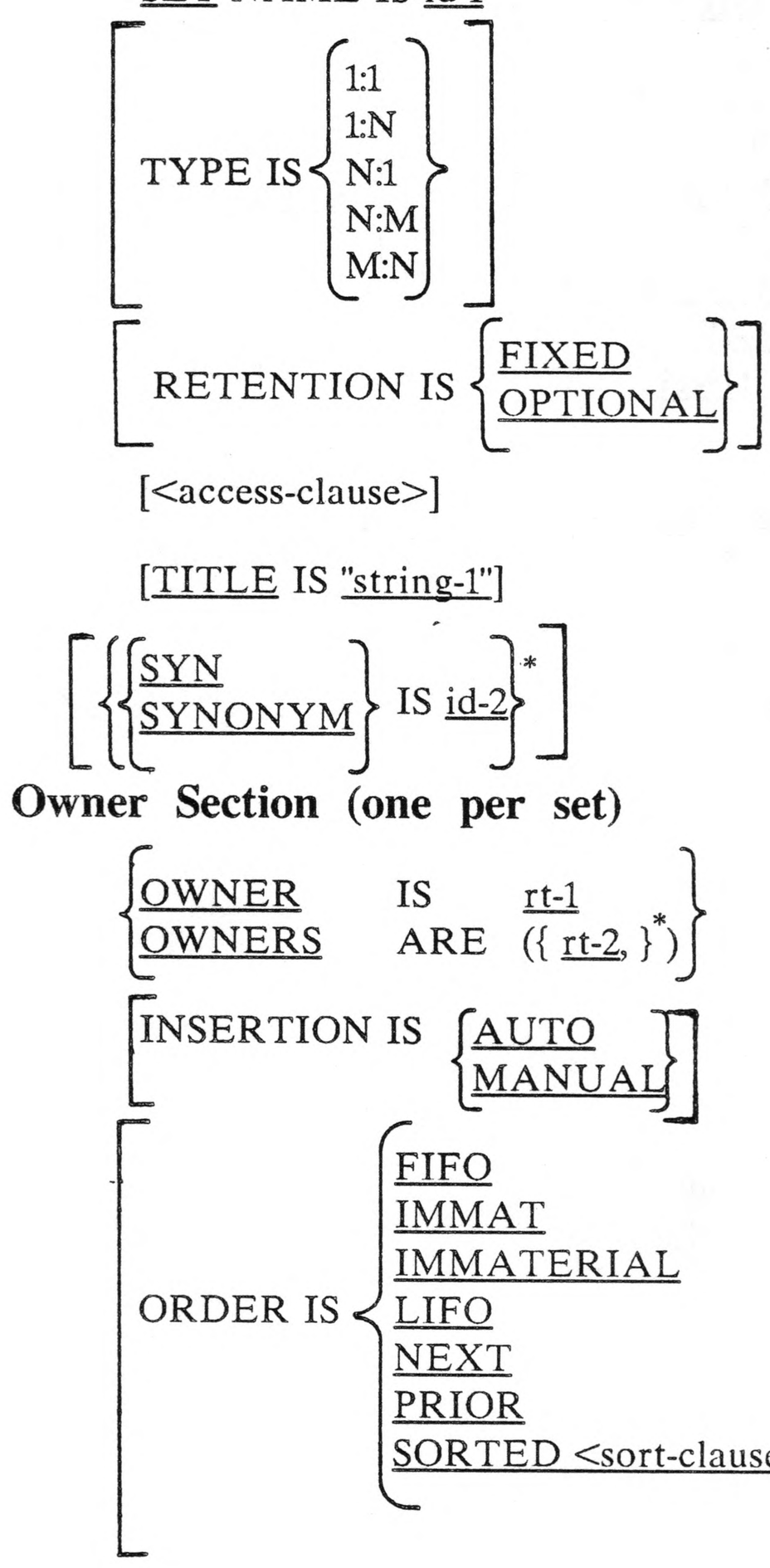
[RANGE IS {lbound / LOWEST} TO {ubound / HIGHEST}]

[{{SYN / SYNONYM} IS id-2}*]

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Description Language Syntax

Set Section (zero, one, or more per schema)



© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Data Description Language Syntax

Member Section (one per set)

```
{ MEMBER   IS    rt-1          }
{ MEMBERS  ARE   ({ rt-2, }*)  }

[ INSERTION IS { AUTO   } ]
               { MANUAL }

[ ORDER IS { FIFO                    } ]
           { IMMAT                   }
           { IMMATERIAL              }
           { LIFO                    }
           { NEXT                    }
           { PRIOR                   }
           { SORTED <sort-clause>    }
```

End Section (one per schema)

```
END [.]
```

© Copyright 1987 Micro Data Base Systems, Inc.

APPENDIX C

MDBS DDL SOURCE SPECIFICATION

LIMITS

APPENDIX C

MDBS.DDL Design Limits

	Minimum	Maximum
Data base element name	1 alphanumeric character	8 alphanumeric characters
User Name	1 alphanumeric character	12 alphanumeric characters
Password	1 alphanumeric character	16 alphanumeric characters
Title string	1 character	determined by line length
Users per data base	1	no limit
Passwords per user	1	1
Access code combinations per data base	1	65535
Areas per data base		
(Version 3a)	1	16
(Version 3c)	1	8
Pages per area*		
(Version 3a)	1	4095
(Version 3c)	1	8191
Page size*		
(Version 3a)	256 bytes	16K (multiple of 256 bytes=area page size)
(Version 3c)	256 bytes	main area page size (up to 16K)
Record types per data base	1	255
Record types per area	0	255
Areas per record type	1	16
Record Occurrences per area	1	no limit*
Record Occurrences per data base	1	no limit*
Data items per record type	0	32767
Data items per CALC-key	1	32767
Data items per sort-key	1	32767

* Subject to an overall data base size of 2^{32} bytes (i.e., over 4 billion bytes) for Version 3a, and in some cases larger for Version 3c.

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
DDL Design Limits

MDBS.DDL Design Limits

	Minimum	Maximum
Data item size		
integer	1	16
real	2	16
binary	1	16374
unsigned	1	16
fixed decimal	1,0	30,7
character	1	16374
string	1	250
Repetitions of a repeating item	1	255
Data item feasibility range	see Table V-1	
Sets per data base	0	no limit
Owner record types per set	1	127
Member record types per set	1	127
Owner record occurrences	0	no limit*
Member record occurrences	0	no limit*

The foregoing are limits imposed by MDBS III software. The environment (hardware, operating system, host language) in which MDBS is used may impose more restrictive practical limits. For instance, if there is insufficient main memory to accommodate pages exceeding 4096 bytes apiece, then a fixed length record occurrence cannot exceed about 4000 bytes (nor can a data item value exceed this size).

* Subject to an overall data base size of 2^{32} bytes (i.e., over 4 billion bytes) for Version 3a, and in some cases larger for Version 3c.

© Copyright 1987 Micro Data Base Systems, Inc.

APPENDIX D

DEFAULTS

MDBS DDL MANUAL
Defaults

APPENDIX D

Optional Omission	Default
Data base (main area) file name	Generated by MDBS; based on data base name
Main area's number of pages	Operating system dependent (typically 50)
Main area page size	Operating system dependent (typically 512 bytes)
Area file name	Generated by MDBS; based on area name
Area's number of pages	Operating system dependent (typically 50)
Area page size	Page size for main area
Record location	Any area
CALC-key duplicates	Allowed
Data item replication factor	1
Data item encryption	No encryption
Data item feasibility range	
(excludes date data item type)	
(lower bound)	LOWEST
(upper bound)	HIGHEST
Set type	l:N
Set retention	Optional
Set insertion for owner	Manual
Set insertion for member	Manual
Set owner order	Immaterial
Set member order	Immaterial
Sort-key duplicates	Immaterial
Sort-key index width	Determined by MDBS (does not exceed sort-key length)
Sorting sequence	English
Title	No title
Synonym	No synonym
Read access	
(record type, set)	a
(data item)	Read access of the data item's record type
Write access	
(record type, set)	a
(data item)	Write access of the data item's record type

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

APPENDIX E

DATA ITEM RANGE EXAMPLES

MDBS DDL MANUAL
Data Item Range Examples

APPENDIX E

These are examples of the maximum range for various types and sizes of data items. The ranges have been calculated in accordance with the maximum range formulas given in Table II-1.

Item Type	Size	Maximum Range (inclusive)	
integer	1	-128	127
integer	2	-32768	32767
unsigned	1	0	255
unsigned	2	0	65535
real	4	$-1.701411 * 10^{38}$ (rounded to 7 digits)	$+1.701411 * 10^{38}$
real	8	$-1.70141183460469 21 * 10^{38}$ (rounded to 17digits)	$+1.70141183460469 21 * 10^{38}$
idec	7	-.99999999 (10^{63})	+.99999999 (10^{63})
idec	8	-.99999999 (10^{63})	+.99999999 (10^{63})
binary	1	00000000	11111111 (binary)
date	--	01/01/1799	12/31/1924
time	--	00:00:00	255:59:59

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

APPENDIX F

ESTIMATING DATA BASE SIZES

MDBS DDL MANUAL
Estimating Data Base Sizes

APPENDIX F

The size of a data base is very important to you, the data base designer and application developer. If the data base is too small, it fills up and you need to either modify the size of the data base or transfer the contents of the full data base to a new, larger data base. If the data base is too large, you waste disk space.

Below are two formulas to help you estimate the data base size needed by your application. The formulas help you determine the total data base size and an appropriate page size for your data base. Following the formulas is a legend of the terms they use.

- TOTAL NUMBER OF BYTES USED BY A SINGLE RECORD OCCURRENCE

$$R+DL+4(S'_{SYS}+S_{1:1}+S_{1:N}+T_{1:N}+S_{N:M}+S'_{N:M}(1+A))+8(S'_{1:N}+S'_{N:M})+(4+W)(S_{1:N}+A(S_{N:M})+S_{SYS})$$

Approximate the total number of bytes for all occurrences of this record type by multiplying the number of record occurrences by the result of the above expression. Repeat this calculation for each record type to obtain an approximate overall data base size for your application.

- PHYSICAL SIZE OF A SINGLE RECORD OCCURRENCE

$$DL+R+4(S_{1:1}+S_{1:N}+S'_{1:N}+T_{1:N}+S_{N:M}+S'_{N:M})$$

The physical record occurrence size approximates the size of a record occurrence without set indices. In most cases, the page size of a data base will be at least as large as the largest record, so calculating the physical record size of each record type can help you determine the page size of your data base.

The number of bytes "used" by a record occurrence are not contiguous in memory. The record is stored as two parts, the fixed length part and the variable length part.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Estimating Data Base Sizes

Legend:

R	in Version 3a, R=3; in Version 3c, R=4
DL	record occurrence data length after data compression (in bytes)
S_{SYS}	number of sorted system-owned sets in which this record type participates
S'_{SYS}	number of non-sorted system-owned sets in which this record type participates
$S_{1:1}$	number of 1:1 sets in which this record type participates
$S_{1:N}$	number of non-system-owned sorted 1:N sets in which this record type participates a member
$S'_{1:N}$	number of non-system-owned non-sorted 1:N sets in which this record type participates as a member
$T_{1:N}$	number of 1:N sets in which this record type participates an owner
$S_{N:M}$	number of N:M sets in which this record type participates and which sorts the occurrences of this record type
$S'_{N:M}$	number of N:M sets in which this record type participates which does not sort the occurrences of this record type
A	average number of records related to an occurrence of this record type via an N:M set
W	average index width (in bytes) of all sorted set indices involving this record type

NOTES:

- Each page contains 8 bytes that cannot be used to store record occurrences.
- Each area reserves a few pages as system pages; these pages cannot hold record occurrences. MDBS.DDL reports the number of system pages in each area immediately after initializing a data base.
- The smallest contiguous block of free space permitted on a page is 4 bytes.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

APPENDIX G

LIST OF QRS KEYWORDS

MDBS DDL MANUAL
QRS Keywords

APPENDIX G

This list contains keywords of the MDBS Query Retrieval System (QRS). Using these keywords in your DDL source specification as data base element names could impair the use of your data base with QRS or other MDBS software modules.

AND
BY
BYE
CLOSE
COMPUTE
DATFORM
DBSTAT
DEFINE
DISP
DISPLAY
ECHO
EDIT
EQ
FOR
FROM
GE
GT
HELP
IN
LE
LIST
LT
NE
NOT
OPEN
OR
QUIT
READ
SELECT
SET
SPEW
STATS
THRU
WRITE
XOR

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

APPENDIX H

ALTERNATIVE LAYOUT FOR DDL SPECIFICATION OF FIGURE III-2

MDBS DDL MANUAL
Alternate DDL Specification Layout

APPENDIX H

```
/*****
 *
 *      Sample Data Base Description (advanced level)
 *
 *****/

        db JOBS, file "JOBS.DB"
                size 300 pages, page size 1024
                logfile "JOBS.LOG"

/*****
 *
 *      Default Data Item Sizes
 *
 *****/

        default for unsigned 2
        default for str 50

/*****
 *
 *      User List
 *
 *****/

        user "BOB SMITH"        with GTC        read access (a,b)   write access a
        user ANALYST            with 7778XK4    read (b-e)          write (b-f)
        user "K FERGUSON"       with "tashi"    access (b,p)
        user "D LEHR"           with "smiles"   access (a-p)

/*****
 *
 *      Additional Data Base Areas - increases total storage capacity
 *
 *****/
        area name is job1, file name is "JOB1.DB"
                size is 100 pages, page size is 512
                pointers not allowed

        area JOB2, file "JOB2.DB"
                size 700 pages
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Alternate DDL Specification Layout

```
/* Note:
 *          Version 3A requires data base areas to use the same page size as the main data base
 *          area.  To use this schema with version 3A, just omit the job1 area page size.  MDBS
 *          defaults to the main area page size.
 */

/*****
 *
 *      Record Type Definitions
 *
*****/

      record BIOGRAPH in JOB2
             read access b  write access p

             item LINE              str
                                    occurs 5 times

      record DEPT, in JOBS
             calc key is NAME nodup
             read access b  write access (a,b)

             item DNUMBER           int 1
                                    range is 1 to 42
                                    syn is DNO
             item is NAME           char 12
                                    encrypted
                                    write access b
             item LOCATION          str 35
                                    range "A" to "ZZZZZZ"
                                    syn LOC

    record employee, in JOB1
             key is ID nodup
             read access (a,d)  write access (a,p)

             item ID                char 9
                                    encrypted
                                    range is "0" to "999999999"
             item LASTNAME          str 20
                                    range "Aa" "Zz"
             item FNAME             str 12
                                    range "Aa" "Zz"
             item PASTJOBS          str 25
                                    occurs 3 times
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Alternate DDL Specification Layout

```
            item YTDEARN        idec 7,2
                                encrypted
                                range 0.00   94000.00
                                read access a, write access a

record JOB in any area
        read access b   write access (a,p)

            item JOBCODE        unsigned 1
                                range 1 to 250
            item DESCRIPT       str

record SKILL in (JOBS,JOB2)
        calc key is SKILCODE nodup
        read access (b,d)   write access f

            item SKILCODE       unsigned
                                range 0 to 3000
                                syn SC
            item DESCRIPT       str 55
            item RATING         real 2
                                range 0.0 to 4.0

/****
 *
*      Set Definitions
*
*****/

      set DETAILS, type 1:1, fixed
            read access (b,d)   write access p

            owner EMPLOYEE
            member BIOGRAPH
                  auto

      set FILLEDBY, type 1:n
            read access (a-d)   write access p

            owner JOB
            member EMPLOYEE
                  fifo, auto
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Alternate DDL Specification Layout

```
set HAS 1:n
        read access (a,b)   write access a

        owner DEPT
        member EMPLOYEE
                lifo, auto

set IDEP 1:n,  read access b

        owner SYSTEM
        member DEPT
                auto

set IJOB 1:n,  read access b

        owner SYSTEM
        member JOB
                fifo, auto

set IEMP, type 1:n, retention fixed
        read access d   write access p

        owner SYSTEM
        member EMPLOYEE
                order sorted by ascending (lastname, Fname)
                insertion auto

set NEEDS n:m
        read access b   write access (f,p)

        owner JOB
                sorted za JOBCODE
        member SKILL
                sorted az (RATING,SKILCODE)

set POSSESS, type n:m
        read access (a,d)   write access (f,p)

        owner EMPLOYEE
                order sorted by az ID
        member SKILL

END
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL

INDEX

$SYSSET Set, II-12, II-14
Access clause, IV-4
Access code list, IV-4
Access codes, IV-2
Access language, I-9
Alternative layouts, H-1
Application developer, I-2
Area section, III-14
Assembly language, I-1
Batch Load Facility, I-4, 12
Boolean DML commands, I-10
Calc algorithms, II-19
Calc key values, II-19
Change command, VI-16, 21
Clause, III-1
Clustering, II-19
CODASYL set, II-8
CODASYL, I-20
CODASYL set, II-8
Command line arguments, VI-2, VI-3
Comments, III-7
Complete initialization, VI-24
Data Base Control System, I-3, 4
Data Base Restructuring System, I-4
Data Description Language (DDL), I-1
Data Description Language Analyzer, I-4
Data dictionary, I-9
Data integrity, I-13
Data Interchange File (DIF), I-12
Data item, II-5
Data item feasibility, V-2
Data item range examples, E-1
Data item section, III-21
Data item size, II-15
Data manipulation commands, I-1
Data Manipulation Language, I-4
Data Manipulation System, I-1
Data security features, V-1
Data security, I-2, IV-1
Data sharing, I-2
Data structuring facilities, I-1
Data Structuring Features, I-19
DATFORM QRS command, III-21
DBCNV DML command, III-21
DDL analysis mode, VI-22
DDL Analyzer, V-4
DDL ANALYZER MESSAGES, VII-1
DDL Defaults, D-1
DDL Design Limits, C-1
DDL keywords, III-1, A-1
DDL source specification, III-6

MDBS DDL MANUAL
Index

DDL Syntax, B-1
Defaults, V-4
Default clause, III-11
Delete Command, VI-9
Design Modification Utility, I-4
Designing MDBS data bases, I-1
Duplicate Key Exclusion, V-2
Edit command, VI-15
Encrypted form, IV-1
Encryption clause, III-22
Encryption, I-14, IV-1
Error messages, VII-1
Estimating data base sizes, F-1
Expert system development, I-18
Feasibility Range Specification, V-2
Field-oriented processing, I-9
FIFO, II-10
File clause, III-8
Fixed set retention, V-1
Flat file structures, II-7
Forked relationships, I-6
Free form input, VI-4
Fully qualified file names, III-4
Get command, VI-20
Global Editing, VI-20
GURU inference engine, I-2
Identification section, III-6, 8
Identifier, III-1
IMMATERIAL, II-11, III-28
Inference engine, I-18
Initialize a data base, VI-23
Initialize a short form, VI-3
Initialize an "OEM" form, VI-3
Input limit, VI-3
Insert command, VI-8
Integers, III-4
Integrity checking, V-1
Interactive commands, VI-6
Interactive Data Manipulation Language, I-4, 11
Internal decimal (IDEC), II-16
INVOKING MDBS.DDL, VI-2
Keyword, III-1
KnowledgeMan/2, I-2
Language clause, III-12
LIFO, II-10
Line editing mode, VI-14
List command, VI-10
Locking techniques, I-15
Log file clause, III-10
Logical structure (schema), I-6, II-5
Many-to-many relationship, I-6, II-8

© Copyright 1987 Micro Data Base Systems, Inc.

MDBS DDL MANUAL
Index

Many-to-one relationship, II-8
Maximum day allowed, V-2
MDBS Version 3a, I-1
MDBS Version 3c, I-1
Member order, II-10, 11
Member record type, II-8
Member section, III-29
Moving Through Text, VI-15
Multiple Member Forked Sets, II-13
Multiple Owner Forked Sets, II-14
Multiuser environments, I-15
Numeric ordering, V-4
Occurs clause, III-22
OEM (original equipment manufacturer), VI-24
OEM initialization, VI-24
One-to-many relationship, I-6, II-8
One-to-one relationship, I-6, II-8
Optional command line arguments, VI-2
OPTIONAL set retention, II-15
Owner order, II-10
Owner record type, II-8
Owner section, III-26
Page clause, III-9
Page initialization message, VI-2
Page size clause, III-9
Page, II-1
Password, III-5, IV-1
Performance, II-4
Physical Validation/Repair Utility, I-14
Placement of records, II-18
Pointer control clause, III-16
QRS keywords, III-1, G-1
Query Retrieval System, I-4, 12
Range Clause, III-23
Read command, VI-12
Record location clause, III-18
Record occurrence, II-6
Recovery and Transaction Logging, I-3
Recovery Utility, I-4
Recursive relationships, I-6
Recursive sets, I-19, II-13
Renumber command, VI-11
Report Definition Language, I-12
Report Definition Language Analyzer, I-4
Retention clause, III-25
Reverse sequence accessing, II-10
Security control, I-14
Selective regeneration, I-13
Set insertion clause, II-15, III-26
Set occurrences, II-9
Set retention, II-15
Set sections, III-24
Set type clause, III-24
Short form, VI-23
Single user DMS buffer size, VI-24
Sort key, II-11
Sort-clause, III-27
Sorted system-owned set, II-12
Special Keys, VI-5
Stop command, VI-25
Stop editing command, VI-19
Strings, III-5
Synonym clause, III-10
System default, II-19
System pages, II-1
System-owned set, II-12
Text Entry, VI-7
Text Insertion, VI-8
The DDL Analyzer, I-3
Time data item, V-2
Title clause, III-10
Transaction logging, I-13
User name, III-5
User section, III-13
Write command, VI-13

© Copyright 1987 Micro Data Base Systems, Inc.

DOCUMENTATION COMMENT FORM

MDBS Document Title:_____

We welcome and appreciate all comments and suggestions that can help us to improve our manuals and products. Use this form to express your views concerning this manual.

Please do not use this form to report system problems or to request materials, etc. System problems should be reported to MDBS by phone or telex, or in a separate letter addressed to the attention of the technical support division. Requests for published materials should be addressed to the attention of the marketing division.

Sender:

_____ _____
(name) (position)

_____ _____
(company) (telephone)

_____
(address)

_____
(city, state, zip)

COMMENTS:

Areas of comment are general presentation, format, organization, completeness, clarity, accuracy, etc. If a comment applies to a specific page or pages, please cite the page number(s).

_____

_____

_____

_____

_____

_____

_____

_____

© COPYRIGHT 1986 Micro Data Base Systems, Inc.